ESPSS

European Space Propulsion System Simulation

EcosimPro Libraries User Manual

(VOLUME 2)

Doc.: 4000103800/11/NL/CP – TN4130	
ESPSS Version: 3.0.beta	Date: 29-09-2013

Client :	ESA
Contract:	ESPSS-3: 4000103800/11/NL/CP
Main Contractor	Empresarios Agrupados (Jose Moral)
Contributors	Astrium / Cenaero / KopooS / VKI

ABSTRACT:

The European Space Propulsion System Simulation (ESPSS) is an ESA initiative that aims to create a European simulation platform for spacecraft and launch vehicle propulsion systems under transient and steady (design & off-design) conditions.

This document contains the User Manuals for the ESPSS-3 Libraries (FLUID_PROPERTIES, FLUID_FLOW_1D, TANKS, TURBO_MACHINARY, COMB_CHAMBERS, *STEADY,* COMP_DATABASE, SATELLITE and ELECTRICAL PROPULSION)provided with ***EcosimPro 5.0 and 5.2.***

Separate chapters are dedicated to each Library and contain an overview (recommendations and tips), a detailed description of the components and the most important aspects of the physical modeling related to each one of the components. Chapter 9 is devoted to demonstrating ESPSS capabilities when coupled to external software. Two applications are selected: zooming and system optimization.

ESPSS-3 includes includes new features as the simulation of solid/hybrid combustors, ramjets, scramjets and the convection/mixing of combusted gases downstream a chamber. It also includes new libraries for steady/quasi-steady models: The STEADY Library designing rocket engine cycles, the SATELLITE Library for orbital and attitude motion and the ELECTRICAL_PROPULSION Library. New useful tips and application examples have been added to help the user define new models.

INDEX

See Volume 1

12. COMP_DATABASE LIBRARY

12.1 OVERVIEW

A generic basis has been specified allowing the creation of an component database compatible with ESPSS for characterizing existing hardware components (e.g. valves, filters, pressure regulators….). The organization of the database is intuitive such that both the user(s) and companies can easily introduce a new ESPSS model for any of their hardware components they provide.

As it is described in the Chapter 5**iError! No se encuentra el origen de la referencia.**, all the ESPSS components are classified in two main groups, capacitive and resistive components, depending on whether they integrate the mass and energy conservation equations, or the momentum equation (computing the mass flow) , respectively. Hence, as a basis on which to build the components' database, a generic abstract capacitive component, and a generic abstract resistive component, are included in the library COMP_DATABASE, and described in section "Abstract components".

In the second section it is explained how to create customized components from the abstract components. These new components are stated in such a way that they have different behavior depending on the "hardware model" chosen from a list. Two examples are described (and included in the library COMP_DATABASE), a customized valve and a simplified tank, which can be used as templates to create new more complex, customized components.

A second approach to create customized components is by inheriting directly from FLUID_FLOW_1D components. Four examples (a customized filter, a customized valve-pressure regulator, a simplified thruster, and a customized tee) are given in the third section.

All the component examples, as well as the abstract components and some additional test cases, are included in the COMP_DATABASE library included in the ESPSS. These examples can then be used as templates to create an electronic database of ESPSS models of specific hardware components.

12.2 ABSTRACT COMPONENTS

The basic rules building capacitive/resistive components within ESPSS are described here depending on the component type.

For more details about the port variables names, see the ESPSS User Manual, chapter 5.2.2

12.2.1 Abstract Component "AbsCapa_comp"

This Abstract Component contains the common formulation that should be inherited by any *Capacitive* type component with two or more ports. This formulation concerns the ports to be inserted and the port variables to be defined inside this component.

Below the code of this component indicating the fixed parts and its structure according to the ESPSS FLUID_FLOW_1D library:

```
-- external libraries
USE MATH
USE THERMAL
USE PORTS_LIB VERSION "1.0"
USE FLUID_FLOW_1D
USE FLUID_PROPERTIES

ABSTRACT COMPONENT AbsCapa_comp (INTEGER nports=2)
"Typical base of a capacitive component calculating pressure, temperature, state vars, etc"

PORTS
      IN  fluid(burnerGasesOption = noBurnGases) f[nports]  CARDINALITY 1  "Inlet/Outlet fluid ports"
```

```
DATA
      REAL Vo = 1            "Tank volume (m^3)"
REAL L = 0.       "Volume length. L=0, volume is a sphere. L<0, horizontal cylinder (m)"
      REAL Pw = 0.          "Wetted perimeter - if zero, the cross area will be assumed circular (m)"
      REAL z_bottom = 0.  "Elevation at the bottom of the volume relative to a z fixed axis (m)"

DECLS
      REAL visc          "Viscosity of the fluid (Pa*s)"
      REAL h             "Mixture -gas and liquid- total enthalpy (J/kg)"
      REAL P             "Pressure at the top of the fluid volume including gravity effects (Pa)"
      REAL T             "Mean temperature of the fluid volume (K)"
      REAL rho           "Mixture -gas and liquid- density of the fluid volume (kg/m^3)"
      REAL vel           "Average fluid velocity (m/s)"
      REAL x_nc          "Non-condensable mass fraction relative to the total fluid mass (-)"
      REAL vsound        "Speed of sound in the fluid (m/s)"

      REAL A             "Fluid cross area (m2)"
      REAL A_wall =  1   "Wall wet area (m^2)"
      REAL Dh            "Hydraulic diameter (m)"

      REAL z_top         "Elevation at the top of the volume relative to a z fixed axis (m)"

PRIVATE INTEGER         n_fluid
      ENUM FluidKeys            fluid
ENUM PerfectGas_Fluids          fluid_nc
      INTEGER  ier              "Error index of thermodynamic function calls (-)"

CONTINUOUS
      -- Geometrical calculation
      Geom_Vol(Vo, L, Pw, Dh, A, A_wall)
      z_top = z_bottom + Vo/A

       -- fluids are the same at any side of the volume
EXPAND_BLOCK (j IN 1, nports)
         f[j].fluid    = fluid
         f[j].fluid_nc = fluid_nc
f[j].n_fluid  = n_fluid
END EXPAND_BLOCK

-- P/T/h/visc/vsound, etc calculations: to be inserted in the inherited components
      -- (see for example the Tank_poly component)
      -- ...

      --  Properties typically calculated in a capacitive component are transferred to the outlet ports
EXPAND_BLOCK (j IN 1, nports)
         f[j].rho   = rho
         f[j].x_nc  = x_nc
          f[j].xd_nc  = 0
f[j].h = h + GRAV*(z_top-f[j].z_jun)
         f[j].P = P + GRAV*(z_top-f[j].z_jun)*rho
f[j].T = T
         f[j].v = vel
         f[j].A = A
f[j].I = 0.5 * Vo/A**2

         f[j].visc  = visc
         f[j].Gcrit = FL_Gcrit_fun(fluid, P, rho, T, 0, vel, vsound, ier)

          EXPAND(i IN noBurnGases) f[j].x_eq[i] = 0
END EXPAND_BLOCK

END COMPONENT
```

12.2.2 Abstract Component "AbsResist_comp"

This Abstract Component contains the common formulation that should be inherited by any *Resistive* type component with two ports. This formulation concerns the ports to be inserted and the port variables to be defined inside this component.

Below the code of this component indicating the fixed parts and its structure according to the ESPSS FLUID_FLOW_1D library:

```
-- external libraries
USE MATH
USE THERMAL
USE PORTS_LIB VERSION "1.0"
USE FLUID_FLOW_1D
USE FLUID_PROPERTIES
-----------------------------------------------
ABSTRACT COMPONENT AbsResist_comp
"Typical base of resistance component calculating mass flow"

PORTS
      OUT fluid f1  (burnerGasesOption = noBurnGases)    "Inlet/Outlet fluid port number 1"
      OUT fluid f2   (burnerGasesOption = noBurnGases)  "Inlet/Outlet fluid port number 2"

DATA
     REAL x_jun = 0         "Junction X coordinate relative to a body axis system (m)"
     REAL y_jun = 0         "Junction Y coordinate relative to a body axis system (m)"
     REAL z_jun = 0         "Junction elevation relative to a body axis system (m)"
     -- other needed data ...

DECLS
     REAL m                "Mass flow - positive from f1 to f2 (kg/s)"

CONTINUOUS

      -- fluids are the same at both sides of the junction
     f1.fluid    = f2.fluid
f1.n_fluid  = f2.n_fluid
f1.fluid_nc = f2.fluid_nc

-- Geometry
     f1.x_jun = x_jun
     f2.x_jun = x_jun
f1.y_jun = y_jun
     f2.y_jun = y_jun
f1.z_jun = z_jun
     f2.z_jun = z_jun

--  Pressure drop eq.: to be inserted in the inherited components
      -- (see for example the Valve_custom component)
      -- ...

      -- enthalpy, volumetric flow & non-condensable mass flows calculation (possible reverse flow)
      f2.mh = f2.m * donor_cell(f2.m, f1.h, f2.h)
      f2.Q  = f2.m / donor_cell(f2.m, f1.rho, f2.rho)
      f2.m_nc = f2.m * donor_cell(f2.m, f1.x_nc, f2.x_nc)
      f2.md_nc = f2.m * donor_cell(f2.m, f1.xd_nc, f2.xd_nc)

     -- Conservation of mass & energy
     f2.m  =  m
     f1.m  = -m
     f1.m_nc =-f2.m_nc
     f1.Q  = -f2.Q
     f1.mh = -f2.mh
      f1.md_nc =-f2.md_nc
```

```
    -- Chemicals mass flows supposed to be 0
    EXPAND(i IN noBurnGases) f2.m_nx[i] = 0
    EXPAND(i IN noBurnGases) f2.m_nx[i] =-f1.m_nx[i]

END COMPONENT
```

12.3 INHERITED COMP_DATABASE COMPONENTS

The basic rules building inherited components within ESPSS are described in this section, depending on the component type (flow calculation in Valves, Tank behabiour, simplified thrusters, etc.)

12.3.1 Component "Tank_poly"

This Component is an example of a simplified Tank behaving according to a politropic expansion or compression:

$$T = T_o \cdot (\rho / \rho_o)^{n-1}$$

Where "n" is the politropic exponent. Below is included and commented the code closing the formulation of the abstract capacitive component "AbsCapa_comp":

```
-- Required Libraries
USE MATH
USE PORTS_LIB
USE FLUID_PROPERTIES

ENUMTankModel = {Userdef_Tank, TMod_001, TMod_002, TMod_003}

COMPONENTTank_polyIS_AAbsCapa_comp
"Tank model according to a politropic exponent"

DATA
    REAL P_o =  100000.    "Initial Pressure (Pa)"
    REAL T_o =  293.15     "Initial temperature (K)"
REAL x_nco = 0.          "Initial non-condensable mass fraction (-)"
REAL npol = 1      "Politropic exponent (-)"

DECLS
    REAL mass           "Total mass in the fluid volume (kg)"
    REAL mass_nc         "Non-condensable mass in the fluid volume (kg)"
    REAL m_tot_in        "Inlet net mass flow (kg/s)"
    CLOSE nports = 2
    DISCR REAL rho_o       "Initial mass in the tank (kg/m^3)"


INIT
     rho_o = FL_prop_vs_pT(fluid, P_o, T_o, fprop_density, ier)
    mass  = rho_o * Vo
    mass_nc  = rho_o * Vo * x_nco

CONTINUOUS
    -- assumed perfect mixture of non-condensable gases
     x_nc = mass_nc/mass

    --  Conservation of Mass
    mass' = SUM(j IN 1, nports; f[j].m)
    mass_nc' = SUM(j IN 1, nports; f[j].m_nc)

    --  Energy eq. (polytropic exponent)
rho = mass / Vo
```

```
T   = T_o * (rho/rho_o)**(npol-1)

      -- thermo function calls
      h  = FL_prop_vs_rhoT(fluid, rho, T, fprop_enthalpy, ier)
       P  = FL_prop_vs_rhoT(fluid, rho, T, fprop_pressure, ier)
       visc   = FL_prop_vs_rhoT(fluid, rho, T, fprop_viscosity, ier)
      vsound = FL_prop_vs_rhoT(fluid, rho, T, fprop_vsound, ier)

      --Average mass flow and main velocity
m_tot_in = SUM(j IN 1, nports; max(0, f[j].m))
vel = m_tot_in / A / rho

END COMPONENT
```

It is remarked the use of the thermo functions (FL_prop_vs_rhoT(…)) allowing to obtain the remaining state variables as a function of the the dynamic state variables, density and temperature.

12.3.2 Component "Valve_cust"

This Component is an example of a CustomizedValve behaving according to some pre-established formulations:

- User defined
- Type 0001
- etc

Below the code of a customized Valve depending on the input data **ValveModel** enumerative variable:

```
USE MATH
USE PORTS_LIB
USE FLUID_PROPERTIES

ENUM ValveModel = {Userdef_Valve, VMod_001, VMod_002, VMod_003}

COMPONENTValve_cust IS_AAbsResist_comp \
"Customized Valve model including sonic flow limitation and variable throat area"

PORTS
      IN  analog_signal (n=1) s_pos     "Valve position signal"

DATA
     ENUM ValveModel Valve_type = Userdef_Valve "Valve type"
REAL Dori= 0   "Valve throat diameter (m)"
REAL Cv= 1     "Valve flow coefficient if Dori=0"
TABLE_1D zeta_vs_pos = {{ 0, 1}, { 1.7, 1.7}} "Pressure drop coef vs. non dimensional stroke position (-)"
     TABLE_1D Arel_vs_pos = {{0,1}, {0,1}}"Non dimensional valve flow area vs. non dimensional stroke position(-
     )"
    REAL tao =   1e-3 "Time constant of the valve actuator as a first order transfer function (s)"

DECLS
      REAL pos_com   "Dimensionless commanded valve position (-)"
      REAL pos_mod   "dimensionless commanded valve area (-)"
      REAL pos       "dimensionless actual valve area (-)"
REAL zeta            "Pressure drop coefficient (-)"
ALG REAL G  = 10      "Mass flow per unit area (kg/s m^2)"
      ALG REAL dP = 1000     "Inlet-outlet pressure drop (Pa)"
      REAL Gst             "Steady unchoked mass flow per unit area (kg/s m^2)"
      REAL Glam             "Laminar mass flow per unit of area (kg/m^2 s)"
INTEGER ier          "Thermo function error code"
REAL rho_up, Gcr_up, Ao
      REAL vel
```

```
INIT
G = 0  -- assumed initial mass flow = 0
pos = 0

CONTINUOUS
     Ao = ZONE(Dori != 0) PI*Dori**2/4 OTHERS Cv*24e-6 * sqrt(2/zeta)

-- Reverse flow allowed (calculates de actual upstream conditions depending on the flow sign
     rho_up = donor_cell(f2.m, f1.rho, f2.rho)
     Gcr_up = donor_cell(f2.m, f1.Gcrit, f2.Gcrit)

-- commanded stroke position
pos_com = max(0, min(1, s_pos.signal[1]) )

-- non-dimensional flow area vs stroke position
 pos_mod = ZONE(Valve_type == Userdef_Valve)     linearInterp1D(Arel_vs_pos, pos_com)  -- user defined
ZONE(Valve_type == VMod_001)         pos_com**3                        -- EqualPercentage valve
ZONE(Valve_type == VMod_002)         pos_com                          -- linear
            -- ...
            -- add as many options as defined in the ValveModel declaration
OTHERS                          pos_com

-- pressure drop coef. vs stroke position
zeta =ZONE(Valve_type == Userdef_Valve)     linearInterp1D(zeta_vs_pos, pos_com)  -- user defined
ZONE(Valve_type == VMod_001)         1.7                                -- put here specific formulation
ZONE(Valve_type == VMod_002)         zeta_cal(f1.A, f2.A, Ao*pos, G, 1, 1)  -- Idelchik
-- ...
            -- add as many options as defined in the ValveModel declaration
OTHERS                          1

-- Actuator simulated as a delay
pos' = (pos_mod - pos)/tao -- - tao2*pos''

-- laminar & unchoked mass flow calculation
     Glam = 0.5*(f1.visc+f2.visc) * 200 /sqrt(Ao) -- it is supposed that laminar flow takes place under Re = 200
Gst  = ssqrt(2*f1.rho*dP)

dP = f1.P  - f2.P
-- Pressure drop equation including critical flow limitation;
            (f1.I+f2.I)  *  (G'*Ao*pos  +  G*Ao*pos')  +  G'*sqrt(Ao)  =  dP  -  0.5  *  max((Gst/Gcr_up)**2,  zeta)  *
                                                                                    fpow(G,Glam,2) / rho_up

m = G * Ao * pos_mod
     vel = G/(rho_up)
END COMPONENT
```

The previous code shows clearly where and how to define a particular pressure drop formulation depending on the Valve_type option

12.4 INHERITED FLUID_FLOW_1D COMPONENTS

The basic rules building inherited FLUID_FLOW_1D components within ESPSS are described depending on the component type.

12.4.1 Component "Filter_cust"

The component Filter is an already existing component in the FLUID_FLOW_1D library inherited from the AbstractJunction. The current formulation is:

Momentum equation: $(I_1 + I_2)\cdot\frac{dm}{dt} = P_1 - P_2 - \Delta P_{REF}\left(\frac{m}{m_{REF}}\right)^n \frac{\rho_{REF}}{\rho}\left(\frac{\mu}{\mu_{REF}}\right)^{2-n}$

where,

n is the exponent of the mass flow in the pressure loss equation
I1 , I2 = half inertia of the connected pipe ends 1 and 2, respectively
m = mass flow that circulates through the sensor
mref = reference mass flow
ΔPref = reference pressure loss
ρref, μref = reference fluid density and viscosity
P1, P2 = pressure at ports
ρ = fluid density

Below it is explained how to build a customized filter component depending on the input data **FilterModel** enumerative variable:

```
ENUM FilterModel= {Userdef_Filter, FMod_001, FMod_002, FMod_003}

COMPONENTFilter_cust IS_A AbstractJunction"Filter with adjustable pressure drop equation"

DATA
      ENUM FilterModel Filter_type = Userdef_Filter        "Filter type"
REAL n = 1          "Exponent of the mass flow in pressure loss equation"
      REAL m_ref = 0.01     "Mass flow at reference conditions (kg/s)"
      REAL dP_ref= 1000     "Pressure loss at reference conditions (Pa)"
      REAL P_ref = 1e5      "Reference pressure (Pa)"
      REAL T_ref = 300      "Reference temperature (K)"
      ENUM FluidName fluid = PfGas_Air          "Reference fluid"

DECLS
      REAL dP_loss         "Pressure loss (Pa)"
      REAL visc           "Viscosity (kg/m*s)"
      DISCR REAL visc_ref= 1e-5  "Viscosity at reference conditions (kg/m*s)"
      DISCR REAL rho_ref = 1     "Density at reference conditions (kg/m^3)"
      REAL rho_in              "Inlet density (kg/m^3)"

INIT
      rho_ref = FL_prop_vs_pT(fluid, P_ref, T_ref, fprop_density, ier)
      visc_ref= FL_prop_vs_pT(fluid, P_ref, T_ref, fprop_viscosity, ier)

CONTINUOUS
      A = min(f1.A, f2.A)

      rho_in = donor_cell (m, f1.rho , f2.rho)
      visc   = donor_cell (m, f1.visc, f2.visc)

      --   Pressure drop calculation
      dP_loss =ZONE(Filter _type == Userdef_Filter)   dP_ref * fpow(m/m_ref, 0.05, n) * (visc/visc_ref)**(2-n)
                                                          * (rho_ref/rho_in)-- user defined
--ZONE(Filter_type == FMod_001)          -- put here specific formulation
-- ZONE(Filter_type == FMod_002)
      -- ...
OTHERS0

      --   Momentum Equation
      (f1.I + f2.I + 1e-6 ) * m' = f1.P + 0.5 * f1.rho * f1.v**2 - f2.P - 0.5 * f2.rho * f2.v**2  - dP_loss

END COMPONENT
```

12.4.2 Component "ValvePressRegDown_cust"

The component ValvePressRegDown is an already existing component in the FLUID_FLOW_1D library inherited from the AbstractJunctionLoss.

The current formulation can be changed as follow to account for several formulations:

```
ENUMPresRegModel= {Userdef_PresR, PMod_001, PMod_002, PMod_003}

COMPONENTValvePressRegDown_cust IS_A AbstractJunctionLoss
          (ENUM PresRegModelPresR_type = Userdef_PresR "Filter type")
          "Pressure regulator controlling the flow area as a function of a downstream pressure signal"

PORTS
      IN analog_signal(n = 1) s_pres "Pressure signal"
DATA
      REAL P_open    =  31e5  "Minimum downstream pressure required to open the valve (Pa)"
      REAL P_close   =  34e5  "Downstream pressure that totally closes the valve (Pa)"
      REAL tao =   1e-3       "Actuator characteristic time (s)"
--      REAL tao2=   1e-6       "Second order time constant of the valve actuator (s)"
      TABLE_1D Kg_vs_pos = { { 0,0.12,0.27,0.4,0.5,0.64,0.8,1} ,{ 0,0.2,0.5,0.7,0.8,0.9,0.97,1} } \
                           "Valve area correction vs dimensionless stroke position"
DECLS
      CLOSE meas_out
      REAL pos     "Dimensionless actual valve position (-)"
      REAL pos_com "Dimensionless commanded valve position (-)"
      CLOSE choked_option = TRUE

INIT
      pos = 0

CONTINUOUS
IF(PresR_type == Userdef_PresR) INSERT
       -- Actuator: As a delay of commanded positions, but limiting the stroke speed
       -- Valve effective flow area as a function of the stroke position

pos_com = ZONE (s_pres.signal[1] > P_close)          0.
ZONE (s_pres.signal[1] > P_open)   (s_pres.signal[1] - P_close) / (P_open - P_close)
OTHERS                                1

pos' = (linearInterp1D(Kg_vs_pos, pos_com) - pos)/tao

ELSEIF(PresR_type == PMod_001) INSERT
       -- put here specific formulationusing same variables as before
...
   END IF
A  = Ao * pos
   dA = Ao * pos'

END COMPONENT
```

It is remarked the use of the ENUM construction parameter **PresRegModel** inside the CONTINUOUS block with IF/ELSEIF/INSERT construction. This implies that the formulation will be closed once the partition is created for the model, and the value of the construction parameter cannot be changed during the simulation. In the previous example (Filter_cust), however, the enumerative variable was declared as a DATA variable, and used with the ZONE technique. In this case, the value of the enumerative (the selected 'model' for the filter) could be changed in the experiment.

12.4.3 Component "Thruster_custom"

This component is an example of a customizedthruster using an empirical correlation giving the thrust as a function of the thruster mass flow calculated by the ESPSS components. In this case, the component does not inherit from any other component:

$$thrust: \quad T[N] = A \cdot \sqrt{P_{in}[bar]} + B$$
$$specific\ impulse: \quad I_{sp}[Ns/Kg] = 2.04 \cdot P_{in}[bar] + 2203.5$$
$$mass\ flow\ rate: \quad \dot{m}[Kg/s] = T/I_{sp}$$

```
USE MATH
USE PORTS_LIB
USE CONTROL
USE FLUID_FLOW_1D

COMPONENT Thruster_custom
PORTS
      OUT fluid(burnerGasesOption = noBurnGases) f1
      IN analog_signal s_pos
DATA
      REAL A = 0.02096  "Parameter A in Thrust calculation from T = A * sqrt(P_in) + B"
      REAL B = -7.669   "Parameter B in Thrust calculation from T = A * sqrt(P_in) + B"
      REAL tau_v = 1e-4   "Characteristic time for valve movement (s)"
      REAL tau_c = 1e-4   "Characteristic time for combustion delay (s)"
      REAL z_jun   = 0.                 "Elevation of the junction (m)"

DECLS
      REAL Isp  "Specific impulse (N s/kg)"
      REAL m    "Mass flow (kg/s)"
--      REAL P_in "Total inlet pressure (bar)"
      REAL pos  "Valve position (-)"
      REAL T    "Thrust (N)"
INIT
     pos = 0
     f1.m = 0

DISCRETE
      ASSERT (f1.n_fluid > 0) FATAL "Working fluid non defined in the model"

CONTINUOUS

pos' = (s_pos.signal[1] - pos) / tau_v

T = (A * fsqrt(f1.P, 100)+ B) * pos

Isp = 2.04e-5 * f1.P  + 2203.5

--       m = ZONE (f1.P > 0 AND pos > 1e-6) T/Isp     OTHERS 0
      m = T/Isp

      --   Conservation of mass
f1.m' = -(f1.m + m) / tau_c

      --Geometry
f1.x_jun = 0
f1.y_jun = 0
f1.z_jun = z_jun

      --port eq.
      f1.mh = f1.m * f1.h
      f1.Q  =  f1.m / f1.rho
f1.m_nc = f1.m * f1.x_nc
f1.md_nc = f1.m * f1.xd_nc

END COMPONENT
```

12.4.4 Component "Tee_custom"

Up to now, it has been explained how to create new customized components from scratch or by inheriting from other components. There is a third method, which is by creating a new component with several (connected) simpler components (a topological component) and adding more equations in the CONTINUOUS block.

A customized Tee component is created below, following this method, using three junctions connected to a central volume. The volume and geometry of the Tee is computed depending on the TeeType enumerative variable; the example is ready to simulate Right tees, "Y" tees, a specific model Tee (with already set data), or with user given angles (UsrDef_Tee option), and can be extended by adding new options to the enumerative variable.

```
USE FLUID_FLOW_1D
USE FLUID_PROPERTIES
USE MATH
ENUM TeeModel = {rightTee, YTee, UsrDefTee, TeeModel_001}

COMPONENT Tee_custom "Adiabatic Tee to simulate a joint or a split of two branches"

PORTS
      OUT FLUID_FLOW_1D.fluid (burnerGasesOption = noBurnGases)    f1
      OUT FLUID_FLOW_1D.fluid (burnerGasesOption = noBurnGases)    f2
      OUT FLUID_FLOW_1D.fluid (burnerGasesOption = noBurnGases)    f3

DATA
      REAL D1 = 0.01 "Diameter of run connection 1 (m)"
      REAL D2 = 0.01 "Diameter of branch connection 2 (m)"
      REAL D3 = 0.01 "Diameter of run connection 3 (m)"
      REAL zeta1 = 0.1 "Loss coefficient of run connection 1 (-)"
      REAL zeta2 = 0.8 "Loss coefficient of branch connection 2 (-)"
      REAL zeta3 = 0.1 "Loss coefficient of run connection 3 (-)"
      REAL Re_lam = 2000 "Laminar Reynolds number (-)"
      REAL x = 0 "Tee X coordinate relative to a body axis system (m)"
      REAL y = 0 "Tee Y coordinate relative to a body axis system (m)"
      REAL z = 0 "Reference elevation of the Tee center relative to a z fixed axis (m)"
      ENUM INIT_OPTION init_option = INIT_PT "Option to specify the initial thermodynamic state"
      REAL P_o = 100000 "Initial Pressure (Pa)"
      REAL T_o = 293.15 "Initial temperature (K)"
      REAL x_o = 0 "Initial quality (-)"
      REAL rho_o = 1 "Ratio of initial liquid mass inside the tee divided by the total tee volume (kg/m^3)"
      REAL x_nco = 0 "Initial non-condensable mass fraction (-)"
      BOOLEAN Gcr_ideal = FALSE "TRUE, ideal critical flow expansion, FALSE, empirical gamma correlation (-)"
      ENUM TeeType Tee_type = right_Tee "Tee type"
      REAL iangle[3] = { 0,90,0} "Port angles: 0, frontal inlet/outlet; 90, lateral inlet/outlet (deg)"
      REAL Vo = 1 "User defined Tee volume (m^3)"
      REAL L = 0. "User defined Tee length (m)"

DECLS
      DISCR REAL Voo = 1
      DISCR REAL Loo, iangleoo[3]
      DISCR REAL iangler[3] = { 0,90,0}
      DISCR REAL iangley[3] = { 120,120,120}
      DISCR REAL iangle_Model_001[3] = {0,90,0}
      DISCR REAL L_Model_001 = 0.1
      DISCR REAL Vo_Model_001 = 0.001

TOPOLOGY
      FLUID_FLOW_1D.Junction(
            choked_option = TRUE) Junction_1(
            x_jun = x, -- Non default value.
            y_jun = y, -- Non default value.
```

```
                z_jun = z, -- Non default value.
                Ao = 0.25 * PI * D1 ** 2, -- Non default value.
                zetaf = zeta1, -- Non default value.
                zetab = zeta1, -- Non default value.
                m_o = 0,
                Re_lam = Re_lam,
                Gcr_ideal = Gcr_ideal -- Non default value.
                )
        FLUID_FLOW_1D.Volume3 Volume(
                init_option = init_option, -- Non default value.
                P_o = P_o, -- Non default value.
                T_o = T_o, -- Non default value.
                x_o = x_o, -- Non default value.
                rho_o = rho_o, -- Non default value.
                x_nco = x_nco, -- Non default value.
                Vo = Voo, -- Non default value.
                L = Loo, -- Non default value.
                Pw = 0,
                z_bottom = z, -- Non default value.
                iangle = iangleoo, -- Non default value.
                iside = { 1,2,2})
        FLUID_FLOW_1D.Junction(
                choked_option = TRUE) Junction_2(
                x_jun = x, -- Non default value.
                y_jun = y, -- Non default value.
                z_jun = z, -- Non default value.
                Ao = 0.25 * PI * D2 ** 2, -- Non default value.
                zetaf = zeta2, -- Non default value.
                zetab = zeta2, -- Non default value.
                m_o = 0,
                Re_lam = Re_lam,
                Gcr_ideal = Gcr_ideal -- Non default value.
                )
        FLUID_FLOW_1D.Junction(
                choked_option = TRUE) Junction_3(
                x_jun = x, -- Non default value.
                y_jun = y, -- Non default value.
                z_jun = z, -- Non default value.
                Ao = 0.25 * PI * D3 ** 2, -- Non default value.
                zetaf = zeta3, -- Non default value.
                zetab = zeta3, -- Non default value.
                m_o = 0,
                Re_lam = Re_lam,
                Gcr_ideal = Gcr_ideal -- Non default value.
                )
        CONNECT Junction_1.f1 TO Volume.f[1]
        CONNECT Junction_1.f2 TO f1
        CONNECT Junction_2.f1 TO Volume.f[2]
        CONNECT Junction_2.f2 TO f2
        CONNECT Junction_3.f1 TO Volume.f[3]
        CONNECT Junction_3.f2 TO f3

    INIT
        IF(Tee_type == rightTee) THEN
                FOR(i IN 1,3)
                        iangleoo[i] = iangler[i]
                END FOR
                Loo = (D1+D2+D3)/3
                Voo = PI/4 *(D1**2+D2**2+D3**2)/3 * (D1+D2+D3)
        ELSEIF(Tee_type == YTee) THEN
                FOR(i IN 1,3)
                        iangleoo[i] = iangley[i]
                END FOR
                Loo = 0.5*(D1**3+D2**3+D3**3) / (D1**2+D2**2+D3**2)
```

```
            Voo = PI/4 *(D1**3+D2**3+D3**3)/2
        ELSEIF(Tee_type == TeeModel_001) THEN
            FOR(i IN 1,3)
                iangleoo[i] = iangle_Model_001[i]
            END FOR
            Loo = L_Model_001
            Voo = Vo_Model_001
        ELSE
            FOR(i IN 1,3)
                iangleoo[i] = iangle[i]
            END FOR
            Loo = L
            Voo = Vo
        END IF
    END COMPONENT
```

13. SATELLITE LIBRARY

13.1 OVERVIEW

SATELLITE is an ECOSIMPRO library primarily dedicated to the simulation of the evolutionary behaviour of components in a satellite. However the library has been designed to simulate much more and is able to perform orbital transfer with thrust, orbit control with thrust and effects of Moon and Sun, LEO satellites control with heliosynchronism effects due to the so called J2 coefficient of the Earth polar flatness. The most important features are the following:

- Orbit propagator with or without thrust
- Orbital perturbations managed (J2, Moon, Sun)
- Satellite components
  - Solar arrays with Sun pressure force and power
  - Reaction wheels
  - Thrusters
  - Gravity booms
  - etc.

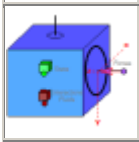
The general design of the components of SATELLITE LIBRARY is coming from open bibliography references.

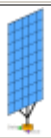
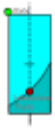
The SATELLITE components can be connected to CONTROL library components in order to perform the attitude control automatically. Specific interface components allow to select the quaternions or the Cardan angles for further process with standard CONTROL components (PID,…) acting on the reaction wheels

13.2 OPERATIONAL COMPONENTS

All the components of the SATELLITE Library are shown in the following table.

Table 13-1 – Components of SATELLITE Library

SYMBOL	COMPONENTS	DESCRIPTION
	Frame_	Non-linear six-degrees-of-freedom satellite body
	GravityBooms_	Array of GravityBooms located from the Centre of Gravity. Only the n first values of the array will be used
	ReactionWheels_	Array of Reaction Wheels or kinetic wheels. Only the n first values of the array will be used
	SensorAttitude_	Measure of attitude from state port Quaternion or CardanAngles, to be used as inputs in a control block

SYMBOL	COMPONENTS	DESCRIPTION
	SensorSpin_	Not used-Not checked-Measure of rotations from state port, to be used as inputs in a control block
	SolarArrays	Array of assemblies BAPTA, Solar cells and structure of Solar arrays. Only the n first values of the array will be used
	Tanks_	Array of Tanks with management of the fluid interactions due to accelerations and rotations. Only the n first values of the array will be used
	Thrusters_	Array of Thrusters of the Orbit and attitude control system. Only the n first values of the array will be used

13.2.1 Enumerative Description:

In order to facilitate the reading of the equations and frame axis, the components of SATELLITE Library relies on 6 ENUM according to the following table.

Table 13-2 – Enumerations list of SATELLITE Library

ENUM ECI	{Xq,Yq,Zq}	Earth Centred Inertial equatorial reference (to Vernal point, ZxX, Earth kinetic momentum) system axis names
ENUM Orb	{X,Y,Z}	Telecom Satellite orbital system (in-plane local horizontal i.e. Velocity for circular, ZxX, to Earth centre) axis names
ENUM Sat	{x,y,z}	Satellite reference system (in plane of launcher i/f --anti-Earth side--, ZxX, normal to launcher i/f) axis names
ENUM Pol	{Xp,Yp,Zp}	Orbital polar system (Focus radius, ZxX, Orbit kinetic momentum) axis names
ENUM FluidAxis	{xf,yf,zf}	Reference frame for fluid free surface under rotation and acceleration, not used
ENUM AttitudeMeasure	{Quaternion, Cardan}	For user's choice in control components

13.2.2 Global variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
g	CONST REAL	9.80665	Standard gravity acceleration	m/s^2

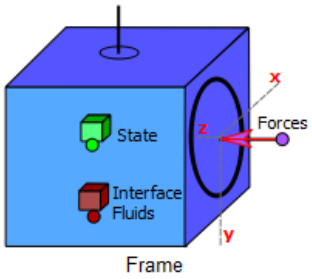
13.2.3 Frame

13.2.3.1 Description

The Frame component performs two main tasks: the orbital dynamic i.e. orbit propagation with or without external forces from thruster(s), with or without perturbations forces from Moon, Sun or Earth flatness (J2) and the Attitude dynamic i.e. the movement around the centre of mass with or without external moments or torques.

13.2.3.2 Orbital Dynamic

The location of the centre of mass with respect to an inertial frame is given by the trajectory integrator that takes into account several forces F coming from the gravitation, the thrusters or the perturbations: $\vec{F} = m\vec{\gamma}$. Because the thrusters are acting with respect to the satellite axis, an orientation change is performed from the satellite axis to the inertial frame before adding the thrust to the vector F above.



13.2.3.3 Construction Parameters

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
nTanks	INTEGER	1		

13.2.3.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	CARDINALITY	DESCRIPTION
S	State		OUT		Output satellite state port
f	Forces		IN		Input external forces port
if	InteractionsFluids	(n4 = nTanks)	IN		Input satellite fluid interaction data port (and allows computing Archimedes pressure for any point of the satellite)

Where the PORT Forces is defined by:

NAME	TYPE	DESCRIPTION	UNITS
F[Sat]	SUM REAL	External forces	N
H[Sat]	SUM REAL	External Angular momentum	N*m*s
M[Sat]	SUM REAL	External moments	N*m
Power	SUM REAL	For components, negative: electrical power used; positive: produced For Frame +electrical power used --or produced if negative--	W
mfr[2]	SUM REAL	For components, negative: mass flow rate used and expelled to outside index 1 for Fuel, 2 for Oxidizer For Frame +mass flow rate used and expelled to outside index 1 for Fuel, 2 for Oxidizer	kg/s

Where the PORT State is defined by:

NAME	TYPE	DESCRIPTION	UNITS
Accel[Sat]	EQUAL REAL	Sat axis acceleration wrt ECI, in Sat axis	m/s^2
JD	EQUAL REAL	Julian Day	day
OCOM[Sat]	EQUAL REAL	Satellite COM location wrt design reference frame at	m/s^2

NAME	TYPE	DESCRIPTION	UNITS
		point O -- i.e. centre of launcher interface--	
PZZZQ	EQUAL REAL	Pilot Date of day from 1900 and time for quaternion , compatible MS Excel	s
Q[4]	EQUAL REAL	quaternion to orient ECI to Satellite axis	-
Qsat[4]	EQUAL REAL	quaternion to orient Orbit axis Orb to Sat axis	-
RAANd	EQUAL REAL	PrecessionZA angle Pol wrt ECI equatorial	deg
RSatEarth[Sat]	EQUAL REAL	satellite COM to Earth position vector in Sat axis for Gravity gradient, etc.	m
RSatSun[Sat]	EQUAL REAL	satellite COM to sun position vector in Sat axis for Solar pressure, etc.	m
R[ECI]	EQUAL REAL	Orbital satellite COM position in ECI	m
V[ECI]	EQUAL REAL	Orbital satellite COM velocity in ECI	m/s
Vsat[Sat]	EQUAL REAL	Orbital satellite COM velocity, in Sat axis	m/s
Wdotd[Sat]	EQUAL REAL	Satellite axis angular acceleration	deg/s^2
WrotSatd [Sat]	EQUAL REAL	Sat axis angular velocity wrt Orb frame, in Sat axis	deg/s
Wrotd[Sat]	EQUAL REAL	Body Sat frame angular velocity wrt ECI, in Sat axis	deg/s
incd	EQUAL REAL	NutationU angle Pol wrt ECIequatorial Inclination of the orbit	deg
mass	EQUAL REAL	Satellite actual mass	kg
omPHId	EQUAL REAL	Rotation propre angle Pol wrt ECI equatorial	deg
phirolld	EQUAL REAL	Roll angle Sat wrt Orb	deg
psiyawd	EQUAL REAL	Yaw angle Sat wrt Orb	deg
thetapitchd	EQUAL REAL	Pitch angle Sat wrt Orb	deg

Where the PORT InteractionsFluids SINGLE IN is defined by:

Construction Parameters of the port InteractionsFluids

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
n4	INTEGER	1	Number of tanks	-

Variables of the port InteractionsFluids

NAME	TYPE	DESCRIPTION	UNITS
FluidMassComponent[n4]	EQUAL REAL	this mass is to be used for the S/C COM computations, even if this is known already in the S/C frame --to avoid unwanted loops-- --and for eventual damping added--	kg
FluidRho[n4]	EQUAL REAL	rho in the tank	kg/m3
ForceArchimedes[16]	EQUAL REAL	Force due to rotation and acceleration: omega, Ai, sinalpha, cosalpha, OCOM[3], Mom[Sat, Sat] dP=rho f .ds i.e. dP= rho gamma h deltaP= integral dP	N
FreeSurfacePoint[n4,Sat]	EQUAL REAL	for each tank 1 to 4, one point of each free surface -- for which 0=f.dsurf when dsurf connect two points of the surface --	m
InertiaMatrixFluids[Sat,Sat]	EQUAL REAL	Inertia Matrix of the connected fluid components at their global COM	kg*m^2
OCOMFluids[Sat]	EQUAL REAL	Location of the global COM of the connected fluid components wrt the S/C design frame at point O	m
SurfaceIsoDeltaP[4]	EQUAL REAL	value of the iso dP of the free surface for each tank	Pa
mfr[2]	EQUAL REAL	Repeated For components, negative: mass flow rate used and expelled to outside index 1 for Fuel, 2 for Oxidizer	

The 3 ports of the component Frame allow to build a satellite quite complex with multiple connections to the ports except for the port InteractionsFluids that is obviously limited to zero or one component.

13.2.3.5 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
COMaccuracy	REAL	0.01	Mixed parameter to catch changes in the COM location, in mass and inertia	m+kg*1000
DDMMYYYYHHMMSS	STRING	"21032012133045"	Real time of the simulation, default 21/03/2012 13:30:45	-
FlagPerturb	REAL	1	Flag at 0 to inhibit the Orbital perturbation by Moon, Sun...	-
FlagPerturbJ2	REAL	1	Flag at 0 to inhibit the J2 perturbation	-
Ixxyyzz[Sat]	REAL	{3000, 4000, 3000}	Moment of inertia Ixx, Roll axis, DRY Satellite Frozen wrt DRY COM , =m*y²+m*z², etc.	kg*m^2
Ixyyzzx[Sat]	REAL	{0,0,0}	Positive Product of inertia Pxy, DRY Satellite Frozen wrt DRY COM, =+m*x*y, etc.	kg*m^2
OCdry_o[Sat]	REAL	{0,0,1}	location of the DRY Satellite centre of mass C wrt the design reference frame O --i.e. wrt launcher interface at point O--	m
OPoint[Sat]	REAL	0	Given Point where the Archimedes pressure is wanted wrt O --input thruster for example--	m
PHI_o	REAL	0	True anomaly actual angle from perigee	deg
RAAN_o	REAL	0	Right ascension of the ascending node or OMEGA (upper case) angle	deg
altApo_o	REAL	833000	altitude apogee, osculating acc. Kepler	m
altPer_o	REAL	830000	altitude perigee, osculating acc. Kepler	m
inc_o	REAL	97	inclination of the orbital plane wrt equator. Should be > 0.001	deg
massDry_o	REAL	3000	Initial dry mass	kg
om_o	REAL	0	omega(lowercase) or perigee argument	deg
phiroll_o	REAL	0	Initial roll angle around x of Sat axis wrt Orb axis	deg
psiyaw_o	REAL	0	Initial yaw angle around z of Sat axis wrt Orb axis	deg
rollpitchyawdot_o[Sat]	REAL	{0,0,0}	Initial rotation speed of Sat axis wrt Orb axis	deg/s
thetapitch_o	REAL	0	Initial pitch angle around y of Sat axis wrt Orb axis	deg

13.2.3.6 DECLS (with non-default values for inherited component):

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
AccelJ2Orb[Orb]	REAL		Gravity Acceleration due to J2 in ECI, Orb	m/s^2
AccelJ2[ECI]	REAL		Gravity Acceleration due to J2 in ECI, Orb	m/s^2
AccelThrusterSunPress[ECI]	REAL		Non gravity forces applied at the port f acceleration= Forces/mass in ECI, Sat	m/s^2
Accel[Sat]	REAL		Non gravity forces applied at the port f acceleration= Forces/mass in ECI, Sat	m/s^2

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
Alt	REAL		Current altitude of the S/C on its orbit	m
AltApo	REAL		Apogee altitude, osculating acc. Kepler	m
AltPer	REAL		Apogee altitude, idem	m
CMWrot[Sat,Sat]	REAL		Cross product matrix for Wrot	rad/s
COMFlag	REAL	1	flag using mixed parameters to catch changes in the COM location, in mass and inertia	-
COMFlag0	REAL	0	flag reference value, to be updated in discrete part	-
CdryC[Sat]	REAL		vector of displacement to the current COM with fluids	m
DQmat[4,4]	REAL		matrix for the quaternion derive equation	-
Day1900	REAL		date more compatible Excel	day
EarthJ2	REAL	+0.00 10827	J2 the second gravitational zonal harmonic of the Earth due to polar Earth flatness , other value (WGS84) 0.001081874 or (IERS) 0.0010826359	-
EarthR	REAL	63781 37	Equatorial earth radius	m
EarthRot	REAL	86400 - 240 + 4.09	23h56m4.09s Sidereal rotational period of Earth the Earth's rotational period with respect to the stars	s
EciLatituded	REAL		Celestial longitude and latitude of the S/C	deg
EciLongituded	REAL		Celestial longitude and latitude of the S/C	deg
Exc	REAL		orbit eccentricity, idem	-
FluidMassTanks	REAL	0	Mass of fluids in all tanks	kg
GMearth	REAL	39860 05000 00000.	G*MEarth	m^3/s^2
GMmoon	REAL	49020 00000 000.	G*MMoon	m^3/s^2
GMsun	REAL	1.3271 5e+02 0	G*MSun	m^3/s^2
Hsat[Sat]	REAL		satellite Angular momentum H= j w	N*m*s
InertiaMatrixSatFrozenComdry[Sat,Sat]	REAL		inertia matrix of the dry sat wrt the local COM when dry	kg*m^2
InertiaMatrixSatFrozen[Sat,Sat]	REAL		inertia matrix of satellite displaced to the final COM including fluids and gas	kg*m^2
InputDataChangedFlag	REAL	1	flag for geometry input changed --into the experiment for example--	-
InputDataChangedFlag0	REAL	0	flag reference value, to be updated in discrete part	-
JD	REAL		Julian day for orbit input	day
JD_o	REAL		Julian Day	day
MatrixECItoOrb[Orb,ECI]	REAL		Coordinate change matrix such that X[Sat]=M[Sat,ECI].X[ECI], etc.	-
MatrixECItoSat[Sat,ECI]	REAL		Coordinate change matrix such that X[Sat]=M[Sat,ECI].X[ECI], etc.	-
Mom[Sat,Sat]	REAL		various parameters for 'if' port for the Archimedes pressure function	

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
			ComputeDeltaPpoint	
MomentCO[Sat]	REAL		Moment of the non-gravity forces applied at the port f =CO^f.F, =f.F^OC	N.m
NAccel	REAL		norm of the acceleration due to the non-gravity forces, norm of the rotation	m/s^2
NOmeg	REAL		norm of the acceleration due to the non-gravity forces, norm of the rotation	m/s^2
OC[Sat]	REAL	{ 0,0,1}	location of COM C wrt the design reference frame at point O	m
PHI	REAL		angle true anomaly in ECI	rad
Perturb[ECI]	REAL		Gravity perturbing forces in ECI	N
QQ[4]	REAL		Quaternions explicit	-
QQorb[4]	REAL		Quaternions of Orb wrt inertial, of Sat wrt Orb, etc.	-
QQsat[4]	REAL		Quaternions of Orb wrt inertial, of Sat wrt Orb, etc.	-
Q[4]	REAL		Quaternions of the S/C wrt Inertial frame, Q[1] being the real part, Q[2],Q[3],Q[4] the unit vector	-
Qo[4]	REAL		Quaternions of the S/C wrt Inertial frame, Q[1] being the real part, Q[2],Q[3],Q[4] the unit vector	-
Qorb[4]	REAL		Quaternions of Orb wrt inertial, of Sat wrt Orb, etc.	-
Qorbinv[4]	REAL		Quaternions of Orb wrt inertial, of Sat wrt Orb, etc.	-
Qsat[4]	REAL		Quaternions of Orb wrt inertial, of Sat wrt Orb, etc.	-
RAAN	REAL		Angle from ECI to Orb	rad
R[ECI]	REAL		Orb axis position	m
R_module	REAL		Orb axis position	m
R_o[ECI]	REAL	{ 1,0,0}	Initial Satellite position in ECI	m
ReMo[ECI]	REAL		radius from Earth to bodies etc.	m
ReMo_module	REAL		radius from Earth to bodies etc.	m
ReS[ECI]	REAL		radius from Earth to bodies etc.	m
ReS_module	REAL		radius from Earth to bodies etc.	m
RmoonSat[ECI]	REAL		radius from Earth to bodies etc.	m
RmoonSat_module	REAL		radius from Earth to bodies etc.	m
RsunSat[ECI]	REAL		radius from Earth to bodies etc.	m
RsunSat_module	REAL		radius from Earth to bodies etc.	m
TimeDay	REAL	0	Time in days, ref. Excel	day
TorqueCOM[Sat]	REAL		the moment applied at the port Force location but translated with the forces to the COM	N.m
V[ECI]	REAL		Orb axis velocity	m/s
V_module	REAL		Orb axis velocity	m/s
V_o[ECI]	REAL	{ 0,1,0}	Initial Satellite velocity in ECI	m/s
WrotSat[Sat]	REAL		satellite instantaneous angular velocity Sat wrt Orb	N*m*s
Wrot[Sat]	REAL		Instantaneous angular velocity Sat wrt ECI	rad/s

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
cosAlpha	REAL		various parameters for 'if' port for the Archimedes pressure function ComputeDeltaPpoint	
dParchimedes[nTanks]	REAL		Archimedes pressure in a given point: OPoint[Sat] for each tank	Pa
errorRep[4]	REAL		Error report if not null for ParametersToCartesian	-
explPower	REAL		Power Sat, explicit	W
horb[ECI]	REAL		horb=R^V specific Angular momentum of Orb	N*m*s/kg
horb_module	REAL		not used	m
horb_o[ECI]	REAL		not used	-
inc	REAL		Angle from ECI to Orb	rad
mass	REAL		Current total mass	kg
om	REAL		angle argument of perigee in ECI	rad
omPHI	REAL		Angle from ECI to Orb	rad
phiroll	REAL		Roll angle	rad
psiyaw	REAL		Yaw angle	rad
shiftH[ECI]	REAL		not used	m^2/s
shiftH_o	REAL		not used	-
shiftR	REAL		not used	m
shiftV	REAL		not used	m/s
sinAlpha	REAL		various parameters for 'if' port for the Archimedes pressure function ComputeDeltaPpoint	
sma	REAL		semi-major axis length, idem	m
thetapitch	REAL		Pitch angle	rad

13.2.3.7 FORMULATION:

References: RD-48 to RD-59

13.2.3.7.1 Trajectory Integrator

13.2.3.7.2 Galilean frame and relative frames: solution of the 2 bodies problem

$\vec{F} = m\vec{\gamma}$ is true for a galilean time reference frame

Hence the absolute satellite acceleration is

$$\vec{\gamma} = \vec{\gamma}_2 = \frac{d^2\vec{r}_2}{dt^2},$$

Here $\vec{F}$ is the force provided by the focus 1 on the spacecraft 2

$$\vec{F} = \vec{F}_{12} = -G\frac{m_1.m_2}{r_{12}^3}\vec{r}_{12} \quad \text{this gives:} \quad \frac{d^2\vec{r}_2}{dt^2} = -G\frac{m_1}{r_{12}^3}\vec{r}_{12}\,.$$

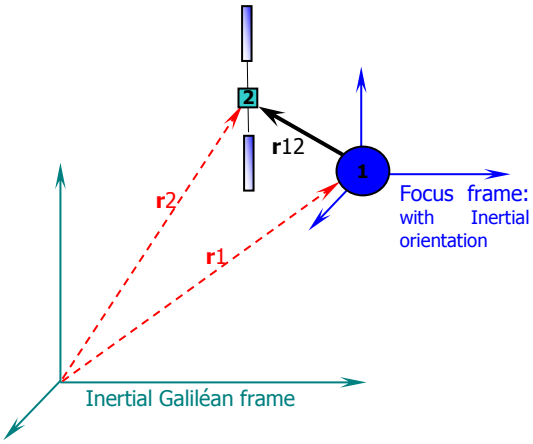


Figure 13-1: Two body problem

Similarly, the force provided by the spacecraft 2 on the focus 1, $\frac{d^2\vec{r}_1}{dt^2} = -G\frac{m_2}{r_{21}^3}\vec{r}_{21}$

We can write a similar equation $\vec{F} = m\vec{\gamma}$ but in the focus reference frame,

with the acceleration, $\vec{\gamma} = \frac{d^2\vec{r}_{12}}{dt^2}$ and $\vec{r}_{12} = \vec{r}_2 - \vec{r}_1$

$$\frac{d^2\vec{r}_{12}}{dt^2} = \frac{d^2\vec{r}_2}{dt^2} - \frac{d^2\vec{r}_1}{dt^2} = -G\frac{m_1}{r_{12}^3}\vec{r}_{12} + G\frac{m_2}{r_{21}^3}\vec{r}_{21}$$

$$\frac{d^2\vec{\mathbf{r}}_{12}}{dt^2} = -\mathbf{G}\frac{(\mathbf{m}_2 + \mathbf{m}_1)}{\mathbf{r}_{12}^3}\vec{\mathbf{r}}_{12}$$

where **G** is the Gravitational constant; and with the derivative performed with respect to the Focus inertial frame.
Of course the satellite mass $\boldsymbol{m_2}$ can be neglected

13.2.3.7.3 Perturbation forces due to the other bodies (Moon, Sun, …)

Note: the user can deactivate the Sun& Moon perturbations in the component by setting the user's data flag (Real) "FlagPerturb " to 0 instead of its default value 1.

As previously, the radius vector from the focus (Earth) to Satellite is : $\boldsymbol{r_{12} = r_2 - r_1}$

$$\frac{d^2\vec{r}_{12}}{dt^2} = -G\frac{(m_2 + m_1)}{r_{12}^3}\vec{r}_{12} - G\frac{m_{moon}}{r_{moon\_2}^3}\vec{r}_{moon\_2} - G\frac{m_{moon}}{r_{1\_moon}^3}\vec{r}_{1\_moon} - etc...$$

central term	moon perturbation	quasi constant term
always managed	when required	when required

The formulation even if called Perturbation represents actually the complete equations for computing a spacecraft trajectory, with or without thrust events, in a solar system.

13.2.3.7.4 Perturbation forces due to the non-spherical earth potential

Note: the user can deactivate the J2 perturbations in the component by setting the user's data flag (Real) "FlagPerturbJ2" to 0 instead of its default value 1.

When only the flatness of the non-spherical earth potential is considered, so called term J_2, the gravitational force per unit of satellite mass ($\frac{\vec{F}}{M}$) in the orbital axis for telecommunication satellites (Z toward Earth, Y = - orbital kinetic moment (also South for GEO satellite in winter) and X= Y ^ Z (^ stand for cross product of vectors)

$$\frac{\vec{F}}{M}(X,\,Y,\,Z) = +\frac{3}{2} J_2 . G.M_e . \frac{\mathrm{Re}^2}{r^4} . \begin{vmatrix} -\sin(2\omega + 2\phi).\sin^2(i) \\ +\sin(\omega+\phi).\sin(2i) \\ -(3.\sin^2(\omega+\phi).\sin^2(i) - 1) \end{vmatrix}$$

where ω is the argument of perigee ; ϕ the true anomaly and i the inclination of the orbital plane with respect to the equatorial plane. This term after coordinates set into ECI frame, is added to the previous sum of perturbation for getting the general force $\vec{\mathrm{Perturb}}$..

13.2.3.7.5 Constants

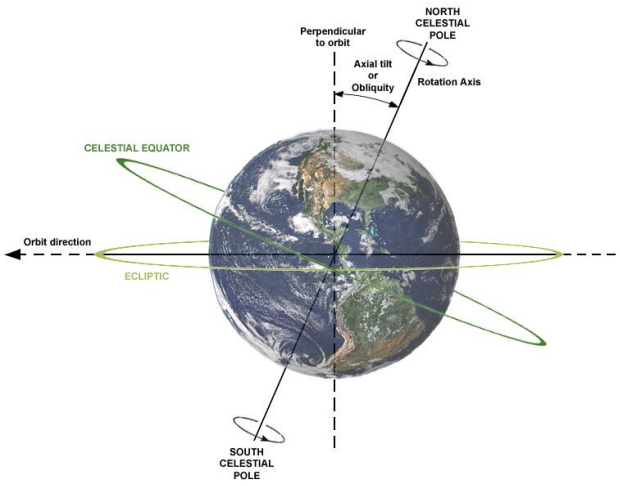
Earth rotation speed = 360 / 86164.0905 °/s (used for the Earth rotating frame, Earth potential computations)

The Astronautical Unit (AU) is a conventional distance between Earth and Sun.

R_{Earth} = 6 378 137 m	$G.m_{Earth}$ = 398 600 500 E+6 m^3/s^2	J_2=+0.0010827 *(no dimension)*
R_{Sun} =695 000 000 m	$G.m_{Sun}$ = 1.32715E+20 m^3/s^2	*AU= 149 597 870 000 m*
R_{Moon} = 1 738 000 m	$G.m_{Moon}$ = 4 902 000 E+6 m^3/s^2	$J_{2,2}$=-1.803E-6 *(no dimension)*

13.2.3.7.6 Frames considered

For the location of the Sun and Moon the ecliptic geocentric frame is considered (the vernal direction γ being the x axis, the normal north of the ecliptic plane being z and y=z ^ x, where ^ designate the vector cross product). This frame is not explicitly used because a further transformation of coordinates is performed for getting the locations in ECI.



For the satellite orbit computations and the J2 perturbations, the equatorial geocentric frame is considered or "Earth Centred Inertial equatorial" with the Enum ECI {Xq,Yq,Zq} (the vernal direction γ being Xq axis, the north of the equatorial plane being Zq and Yq = Zq ^ Xq).

The transformation of coordinates given in the ecliptic geocentric frame to coordinates in the equatorial geocentric frame is coming from a rotation around the vernal direction (which is at the crossing point between the two planes) by the angle obliquity of the ecliptic (23.44 deg).

13.2.3.7.7 Integration

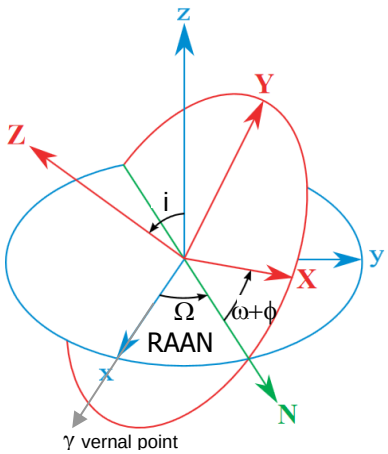
The frame considered for the integration is the Earth Centred Inertial equatorial or ecliptic frame

$$\frac{d\vec{r}_{12}}{dt} = \vec{v}_{12}$$ *Transform second order differential system into one order double size system*

$$\frac{d\vec{v}_{12}}{dt} = -G\frac{m_1}{r_{12}^3}\vec{r}_{12} + \sum \vec{Thrust}/m_2 + \sum \vec{Perturb}/m_2$$ Term ($m_2 + m_1$) rounded to m_1. Sum of vectors

$$\frac{dm_2}{dt} = -\sum \frac{\|\vec{Thrust}\|}{g_0.Isp}$$ Sum of the norms values

The initial conditions are part of the data with the starting date of the computation, the orbit parameter: apogee and perigee altitudes, and the Euler angles as sketched at right: right ascension of the ascending node (RAAN) also usually called angle "Ω", inclination with respect to equator angle "i" (with a possible restriction that "i" should be > 0.001°), perigee argument (i.e. angle from the node axis to the perigee, angle "ω") plus the actual true anomaly (i.e. the angle from the perigee to the actual location, angle "φ").



Note: the force $\vec{Perturb}$ may contain further other perturbation forces, like sun pressure force, other non-spherical potential terms, etc.

13.2.3.8 ATTITUDE Dynamic

The dynamic equation for the attitude of satellite around its centre of mass is given in the references by the derivation of the kinetic moment $\vec{H}$ with respect to an inertial frame $\frac{d\vec{H}}{dt} = \vec{M}$ with $\vec{M}$ the sum of all the external moment with respect to the centre of mass and $\vec{H} = [I]*\vec{\Omega}$ where I is the inertia matrix of the satellite, $\vec{\Omega}$ the instantaneous rotation of the satellite axis with respect to the inertial frame (not related at all to the RAAN angle introduced above paragraph) and with all coordinates and derivative in the inertial frame.

Because the matrix I and most of the external moments coordinates are better available when using the satellite axis, the derivative in the above equation can be performed with respect to the satellite axis but adding the effect of the derivative of the satellite axis with respect to the inertial frame: $+\vec{\Omega}\wedge\vec{H}$ with $\vec{M},\vec{H},\vec{\Omega}$ now all coordinates written in the satellite axis (but still $\vec{\Omega}$ with respect to the inertial frame), and also $[I]$ written in the satellite axis.

This general dynamic equation with all coordinates written in the satellite axis is refined in order to take into account:

1. the mobile parts of the satellite for the control : the reactions wheels, kinetic moment $\vec{H}_{RWi}$ written in the satellite axis
2. the other mobile parts of the satellite: the solar arrays and the fluid in the tanks , kinetic moment $\vec{H}_i$ written in the satellite axis
3. the remaining dry and non-mobile constituents of the satellite that include the without any rotation mobile parts (so called frozen part), with kinetic moment $\vec{H}$ written in the satellite axis

Finally, we get the general form:

$$\frac{d\vec{H}}{dt} + \vec{\Omega} \wedge \vec{H} = \vec{M}_{Control} + \vec{M}_{Perturbation}$$

with $\vec{M}_{Control} = \left\{ M_{thruster} - \sum_i \frac{d\vec{H}_{RWi}}{dt} \right\}$ the torques due to the thrusters and Reaction Wheels (RW)

and $\vec{M}_{Perturbation} = \left\{ M_{perturb} - \sum_i \vec{\Omega} \wedge \vec{H}_{RWi} - \sum_i \vec{\Omega} \wedge \vec{H}_i - \sum_i \frac{d\vec{H}_i}{dt} \right\}$ the perturbation torques due to the other mobile or flexible parts of the satellite (solar arrays and the fluid in the tanks) and with $\vec{\Omega}$ the instantaneous rotation with respect to the inertial orientation frame of the satellite and the rotation with respect to the satellite of the reaction wheels $\vec{\Omega}_{RWi}$ and of the other mobile parts $\vec{\Omega}_i$, but with all their coordinates and derivative written in the satellite axis.

13.2.3.9 ATTITUDE angle Dynamic

According to the previous equations, it is mandatory to know at each time the attitude angles. The general dynamic equation allows the computations of those angles. However some orientation cases may produce singularities in the solution of the equations for the attitude angles.

The attitude angles of the satellite can be given by the Cardan angles (yaw, pitch, roll) of the satellite axis with respect to the orbital frame. The attitude angles of the orbital frame can be given by the Euler angles (precession, nutation, proper rotation) with respect to the inertial frame.

In order to get robustness in the solutions, the equations used to deal with the attitude angles are coming from the quaternions theory.

The time derivative of the quaternions giving the orientation of the satellite with respect to the inertial frame is a well-known relation: $\frac{dQ}{dt} = \frac{1}{2} \cdot Q.\Omega$ in quaternions algebra (see the § QUATERNION's algebra Formulation at the end of this chapter) where Ω is considered as a pure imaginary quaternion with respect to the satellite axis. Translated to the classic algebra with matrix product, it becomes:

$\frac{\mathbf{dQ}}{\mathbf{dt}} = \frac{1}{2}\left[\hat{\Omega}\right] \cdot \mathbf{Q}$ where "Omega hat $\left[\hat{\Omega}\right]$" is a 4x4 matrix deduced from the coordinates in the satellite axis of $\vec{\Omega}$ (instantaneous rotation of the satellite axis with respect to the inertial frame) and Q the quaternion with its real part and its 3 imaginary components in the inertial frame of quaternion (i.e. 4 coordinates) written as a column matrix.

This integration produces at each time "t" the quaternion $Q(t)$ that enable to retrieve (with suited conversion matrixes) the attitude angles without having any risk of singularities because the quaternion considered are unit quaternion.

COM and inertia matrix management

The current COM is coming from the COM of the satellite dry given in the data and from the COM of the fluids if a component Tanks is connected to the Frame *(only zero or one Tanks component is allowed but the Tanks component is an array of up to 4 tanks)*.

The above inertia matrix I is coming from

- the inertia matrix of the satellite dry given in the data wrt to the COM of the dry satellite. It is further displaced to the current COM.
- the inertia matrix of the fluids wrt the COM of the fluids provided at the InteractionsFluids port "if" from the Tanks component. It is further displaced to the current COM of the whole satellite. Without Tanks component that matrix and the fluid mass are zero.

- those two final inertia matrixes wrt the current COM of the whole satellite are added for finally getting the matrix I called in the code "InertiaMatrixSatFrozen" with the word frozen because the reaction wheels inertia due to their rotation are not included here because those mobile parts are managed separately in the equations above.

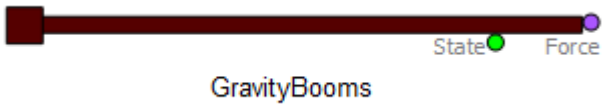
For this release of the library the update of the current COM and InertiaMatrixSatFrozen is almost continuous but by step. This is managed in a discrete part "when" a composite flag called "COMFlag" mixing fluid mass changes plus absolute location of the fluid COM changes plus InertiaMatrixFluids diagonal changes pass over a boundary set with by the user's data "COMaccuracy". The user can trim this user's data for reaching the wanted accuracy of the update. Because the update is managed in the discrete part "when", the update of COM and inertia matrixes are reported in the monitor log.

For this release of the library it is not needed to connect a Tanks component to see the effects of the thruster consumption on the total mass. But in such case, the COM location and the Inertia remain unchanged at their original dry value.

13.2.4 GravityBooms

13.2.4.1 Description

The Gravity Booms is a component array of Gravity Booms. Only the n first values of the array will be used.



13.2.4.2 Construction Parameters

NAME	TYPE	DESCRIPTION	UNITS
n	INTEGER	Number of GravityBooms	

13.2.4.3 PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	CARDINALITY	DESCRIPTION
S	State		IN		satellite state port
f	Forces		OUT		external forces, Moment, power, flow, ... OUT port

13.2.4.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
FlagOnOff	REAL	1	flag set to 0 for no outputs, flag set to 1 for power and forces&moments outputs	-
Floc[3]	REAL	{ 0,0,1}	location of all the booms at the Centre of Mass of the satellite excluding its, wrt the design reference frame of the Satellite --generally launcher interface--	m
Forient[4,3]	REAL	{ { 0,0,1} ,{ 0,0,1} ,{ 0...	axe of the boom, not necessarily normalised	-
Length[4]	REAL	{ 10,10,10,10}	length of the boom; Only the n first values will be used	m2
MassEnd[4]	REAL	{ 1,1,1,1}	mass at the end of the Boom length	kg

13.2.4.5 DECLS:

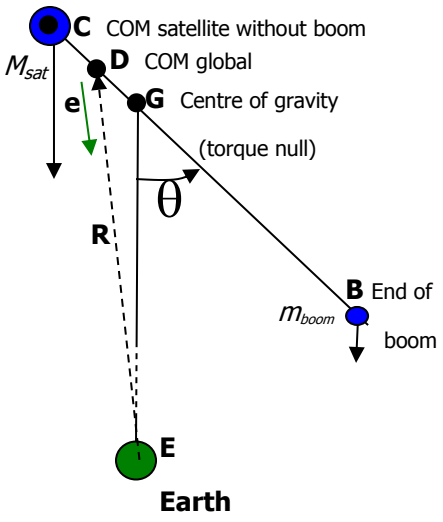
NAME	TYPE	INITIAL	DESCRIPTION	UNITS
GMearth	REAL	398600	G*MEarth	m^3/s^2

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
		500000000.		
InputDataChangedFlag	REAL	1	flag for geometry input changed --into the experiment for example--	-
InputDataChangedFlag0	REAL	0	flag reference value, to be updated in discrete part	-
Mn[n,Sat]	REAL		individual data	-
RSatEarth_module	REAL		distance to Earth	m
TorqueCOM[Sat]	REAL		individual data	-
Torque[n,3]	REAL		individual data	-
crossMatCOMtoMassEnd[n,3,3]	REAL		individual data	-
eAxe[n,3]	REAL		individual data	-
eEarth[3]	REAL		individual data	-
vectorCOMtoMassEnd[n,3]	REAL		individual data	-

13.2.4.6 FORMULATION:

A vector of Gravity Booms is considered.

Each Gravity Boom is characterized by its length, the location of satellite centre of mass COM (a priori excluding the boom) at point C with respect to the geometrical frame of the satellite [O, x,y,z], the orientation of the boom axis $\vec{e}$, the mass at the end of the boom (at point B).



A connection with the State port allows to know the vector "Satellite to Earth" for which the module is *R* and the unit vector is $\vec{e}_{earth}$.

The equations used in the components are

- Mass flow rate: $mfr = 0$
- Force: $\vec{F} = \vec{0}$ for the considered gravity torque.
- Moment wrt O: it is a pure torque, $\vec{M}_O = \vec{M}_D$

  with $\vec{M}_D \approx \frac{3\mu.m_{boom}}{R^3}.(\vec{e}_{earth} \cdot \vec{CB}).\vec{DB} \wedge \vec{e}_{earth}$

  $\vec{M}_D \approx \frac{3\mu.m_{boom}}{R^3}.(\vec{CB} \cdot \vec{e}_{earth})\vec{CB} \wedge \vec{e}_{earth}$

  where μ is the gravitation of earth.

  (according to the sketch, this come from the definition $\vec{M}_G = \vec{0}$

  and computing the moment at point D)

- Angular momentum: $\vec{H} = \vec{0}$
- Power: $Power = 0$

Note: the user can deactivate the outputs of the component by setting the user's data flag (Real) "FlagOnOff" to 0 instead of its default value 1.

Annex for the derivation of the Moment's equation

Notations: the dot $\cdot$ is used for the scalar product of 2 vectors and $\wedge$ for their cross product, The vector between the two points D and C is noted here $\vec{DC}$ instead of the classic $\overrightarrow{DC}$

- The point D is the global centre of mass (COM) defined by $\vec{DC}.M_{sat} + \vec{DB}.m_{boom} = 0$
- The point G is the centre of gravity such that the torque of the forces due to the gravity on each mass is null: $\vec{GC} \wedge \vec{F}_{sat} + \vec{GB} \wedge \vec{F}_{boom} = 0$

The moment at D is $\vec{M}_D = \vec{DC} \wedge \vec{F}_{sat} + \vec{DB} \wedge \vec{F}_{boom}$ with $\vec{F}_{sat} = k_{sat}.\overrightarrow{EC}$ and $k_{sat} = -\frac{\mu}{R^3}.M_{sat}.\frac{1}{EC^3/R^3}$ Note $\frac{EC^2}{R^2} = \frac{ED^2}{R^2} + \frac{DC^2}{R^2} + 2.\frac{\vec{ED}\cdot\vec{DC}}{R^2}$ and $\frac{DC^2}{R^2} \approx 0$ as $D \approx C$, so with $\vec{ED} = -\vec{e}_e .R$ $\frac{EC^2}{R^2} \approx 1 - 2\frac{\vec{e}_{earth}\cdot\vec{DC}}{R}$; $\frac{1}{EC^3/R^3} = 1 + 3\frac{\vec{e}_{earth}\cdot\vec{DC}}{R}$ so:

$k_{sat} = -\frac{\mu}{R^3} M_{sat}\left(1 + \frac{3\vec{e}_{earth}\cdot\vec{DC}}{R}\right)$. Idem for $\vec{F}_{boom} = k_{boom}.\vec{EB}$ and $k_{boom} = -\frac{\mu}{R^3} m_{boom}\left(1 + \frac{3\vec{e}_{earth}\cdot\vec{DB}}{R}\right)$

Simplification $\vec{M}_D = k_{sat}\vec{DC} \wedge \vec{ED} + k_{boom}\vec{DB} \wedge \vec{ED}$ because $\vec{DC} \wedge \vec{DC} = 0$ and $\vec{DB} \wedge \vec{DB} = 0$

And $\vec{M}_D = -\frac{\mu}{R^3}\left(M_{sat}\vec{DC} + m_{boom}\vec{DB}\right) \wedge \vec{ED} - \frac{\mu}{R^3}\left(M_{sat}\frac{3\vec{e}_{earth}\cdot\vec{DC}}{R}\vec{DC} + m_{boom}\frac{3\vec{e}_{earth}\cdot\vec{DB}}{R}\vec{DB}\right) \wedge -\vec{e}_{earth}.R$

Further, $\vec{M}_D = 0 - \frac{\mu}{R^3}.\left(-m_{boom}\frac{3\vec{e}_{earth}\cdot\vec{DC}}{R}\vec{DB} + m_{boom}\frac{3\vec{e}_{earth}\cdot\vec{DB}}{R}\vec{DB}\right) \wedge -\vec{e}_{earth}.R$ thanks to the definition of D $M_{sat}\vec{DC} = -m_{boom}\vec{DB}$ $\vec{M}_D = \frac{3\mu}{R^3}.m_{boom}\left(\vec{e}_{earth}\cdot\left(\vec{DB} - \vec{DC}\right)\right)\vec{DB} \wedge \vec{e}_{earth}$

$\vec{M}_D \approx \frac{3\mu.m_{boom}}{R^3}.(\vec{e}_{earth}\cdot\vec{CB})\vec{DB}\wedge\vec{e}_{earth}$ and with $D \approx C$ one get $\vec{M}_D \approx \frac{3\mu.m_{boom}}{R^3}.(\vec{CB}\cdot\vec{e}_{earth})\vec{CB}\wedge\vec{e}_{earth}$

13.2.6 ReactionWheels

13.2.6.1 Description

The ReactionWheels is a component array of Reaction Wheels. Only the n first values of the array will be used.

13.2.6.2 Construction Parameters

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
n	INTEGER	1	Number of ReactionWheels or kinetic wheels <=4	

13.2.6.3 PORTS:

PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
f	Forces		OUT	external forces, Moment, power, flow, ... port
s	PORTS_LIB.analog_signal	(n = n)	IN	Commanded rpm (rad)

13.2.6.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
ElectricalEfficiency	REAL	0.3	for the conversion electrical to mechanical power	-
Forient[4,3]	REAL	{ { 1,0,0} ,{ 0,1,0} ,{ 0...	direction of the axis of rotation in Sat, not necessarily normalised by default: X Y Z	-
Inertia[4]	REAL	{ 0.1,0.1,0.1,0.1}	Inertia 'm.r²' of the RW wrt to its axis ; Only the n first values will be used	kg.m2
tau	REAL	0.1	time response	s

13.2.6.5 DECLS:

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
InputDataChangedFlag	REAL	1	flag for geometry input changed --into the experiment for example--	-
InputDataChangedFlag0	REAL	0	flag reference value, to be updated in discrete part	-
Powern[n]	REAL		individual data	-
eAxeSatAxis[4,Sat]	REAL		individual data	-
eAxe[4,3]	REAL		individual data	-
jj	INTEGER		counter	-
omega[n + 1]	REAL		at least one dimension to avoid declaration error when setting n==0	rd/s
rpmControl[4]	REAL	{ 0,0,0,0}	the wanted rpm	rpm
rpm[n]	REAL		individual data	-

13.2.6.6 FORMULATION:

A vector of Reaction Wheels is considered.

Each Reaction Wheel is characterized by its inertia, its orientation of the rotation axis $\vec{e}$, the wanted steady state rotation velocity ω_{ss}, the time response ω and the electrical efficiency of the motor η.

The equations used in the components are

- Mass flow rate: $mfr = 0$
- Force: $\vec{F} = \vec{0}$
- Moment wrt O: $\vec{M}_O = \vec{0}$
- Angular momentum: $\vec{H} = I.\omega.\vec{e}$ where $\frac{d\omega}{dt} = \frac{(\omega_{ss} - \omega)}{\tau}$
- Power: $Power = -\frac{I.\frac{d\omega}{dt}.\omega}{\eta}$ the negative value comes from the rule of the Ports direction with type OUT.

13.2.7 SensorAttitude

13.2.7.1 Description

Measure of attitude from state port Quaternion or CardanAngles, to be used as inputs in a control block

SensorAttitude

13.2.7.2 Construction Parameters

None

13.2.7.3 PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s	State		IN	Volume signals
s_out	PORTS_LIB.analog_signal	(n = n_out)	OUT	Outlet signal

13.2.7.4 DATA:

NAME	TYPE	DESCRIPTION	UNITS
Bias	REAL	Bias	-
Gain	REAL	Gain	-
Measure	ENUM AttitudeMeasure	Required measure type	

CLOSED:

PARAM/DATUM/PORT	VALUE
bias	{ 0,0,0}
gain	{ 1,1,1}
n_out	3

13.2.7.5 DECLS:

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
CardanAngles[3]	REAL			
var[n_out]	REAL		Measured variable	-

13.2.7.6 FORMULATION:

According to the value of the ENUM Measure

Measure = Quaternion : Output the imaginary part of the quaternion of the Satellite attitude with respect to the orbital frame (that is to be nullified for an Earth pointing attitude control)

Or

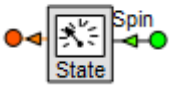
Measure = Cardan : Output the Cardan angles (yaw pitch roll: psi, theta and phi in degrees) (that are to be nullified for an Earth pointing attitude control)

13.2.8 SensorSpin

13.2.8.1 Description

Not used-Not checked-

Measure of rotations from state port, to be used as inputs in a control block



SensorSpin

13.2.8.2 FORMULATION:

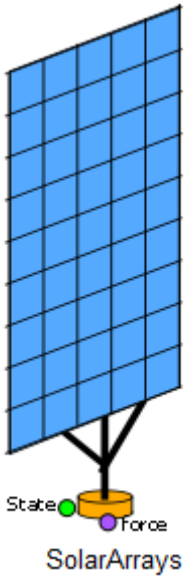
Output the rotation rate (in degrees per second) of the Satellite with respect to the orbital frame

13.2.10 SolarArrays

13.2.10.1 Description

The SolarArrays is a component array of assemblies BAPTA, Solar cells and structure of solar arrays. The case of fixed solar array on the satellite body is managed (like for cubesat solar arrays). Only the n first values of the array will be used.

This component can be thus also be used for the simulation of the antenna reflectors (fixed solar array case) and the effect of the solar pressure on them.



13.2.10.2 Construction Parameters

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
n	INTEGER	1	Number of SolarArrays	

13.2.10.3 PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
S	State		IN	satellite state port
f	Forces		OUT	external forces, Moment, power, flow, ... OUT port

13.2.10.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
AngleProperRotation[4]	REAL	{ 0,0,0,0}	not used, for future enhancements. Best SA orientation wrt Sun is used instead	-
Area[4]	REAL	{ 10,10,10,10}	area; Only the n first values will be used	m2
Automatic	REAL	1	flag Best Orientation To Sun to orient the SA when=1 else when <>1 SA on body fixed	-
Ca	REAL	0.2	absorption impulse coefficient for sun rays 0 to 1-Cs-Cdiffus	-
Cs	REAL	0.7	specular reflexion impulse coefficient for sun rays Ca to 1-Cdiffus	-
FlagOnOff	REAL	1	flag set to 0 for no outputs, flag set to 1 for power and forces&moments outputs	-

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Floc[4,3]	REAL	{ { 0,10,1} ,{ 0,-10,1} ,...	Solar Pressure centre location wrt design reference frame of the Satellite --generally launcher interface--	m
Forient[4,3]	REAL	{ { 0,1,0} ,{ 0,-1,0} ,{ ...	axe of rotation direction, not necessarily normalised	-
canonicalNormalToCells[4,3]	REAL	{ { 1,0,0} ,{ 0,1,0} ,{ -...	used only for Not Automatic=1 with SA on body fixed	-
eta	REAL	0.24	efficiency of the solar cells	-

13.2.10.5 DECLS:

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
AU	REAL	150000000000.	1 AU distance	m
Cd	REAL	2	diffusely reflexion impulse coefficient for sun rays	-
FFn[n,3]	REAL		individual data	-
FlagEclipse	BOOLEAN	FALSE	Flag = True when eclipse occurs, else False	-
Fn[n,Sat]	REAL		individual data	-
FractionSun	REAL	1	0 when in eclipse, 1 in the sun	-
InputDataChangedFlag	REAL	1	flag for geometry input changed --into the experiment for example--	-
InputDataChangedFlag0	REAL	0	flag reference value, to be updated in discrete part	-
Mn[n,Sat]	REAL		individual data	-
MomentCO[Sat]	REAL		torque displacement from Origin of launcher I/F to COM :MomentCO+M/O=M/C	Nm
PCosS[n]	REAL		accessory	-
Powern[n]	REAL		individual data	-
RSatSun_module	REAL		distance	m
SolarPressure	REAL		individual data	N
SunFlux	REAL	1375	at 1 AU, surface normal to the Sun rays	W/m2
TorqueCOM[Sat]	REAL		disturbance torque seen at Satellite COM point C due to sun pressure	Nm
crossMatAxe[n,3,3]	REAL		individual data	-
crossMatOFn[n,3,3]	REAL		individual data	-
eAxe[n,3]	REAL		individual data	-
eInPlane[n,3]	REAL		individual data	-
eNormalBodyFixed[n,3]	REAL		individual data	-
eNormal[n,3]	REAL		individual data	-
eSun[3]	REAL		individual data	-
en_esun[n]	REAL		individual data	-

13.2.10.6 FORMULATION:

A vector of Solar Arrays is considered.

Each Solar Array is characterized by its area, its centre location "S_{centre}" with respect to the geometrical frame of the satellite [O, x,y,z], the orientation of the rotation axis $\vec{e}$ (used only for the case Automatic=1) specified by a not necessarily unit vector in the satellite axis, the efficiency of the solar cells η ·(to be set to zero in case of reflector antenna) , the coefficients of reflection *Cr* and the fraction of specular reflection *Cs* of the solar flux (coefficients of absorption =1-*Cr*; non-specular reflection (i.e. diffuse reflection) =1-*Cs)* with *Cr* and *Cs =0 to 1)*.

A connection with the State port allows to know the vector Satellite to Sun for which the module *R* and the unit vector is $\vec{e}_{Sun}$. The eclipses of sun are able to be computed thanks to the data from the port State as well (a specific function Eclipse(S.RSatSun,S.RSatEarth, FlagEclipse, FractionSun) output continuously for each time a Boolean flag "FlagEclipse" that is used to turn off all the effects of the sun when it is true). Because the Boolean flag is managed in the discrete part, the eclipse events are reported in the monitor log.

The automatic best orientation with respect to the Sun is considered when the data Automatic is set to 1. Else the SA is considered fixed in the satellite frame (case also for the reflector antenna if any).

The equations used in the component (with AU for Astronomical unit) and when the Boolean flag "FlagEclipse" is false, are the following:

- Mass flow rate: $mfr = 0$

Force: $\vec{F} = F_a.\vec{e}_{sun} + F_s.\vec{e}_n + F_d.\vec{e}_d$ where $\vec{e}_n$ is the unit vector of the normal to the solar array given by $\vec{e}_n = -\vec{e}\wedge(\vec{e}\wedge\vec{e}_{Sun})$ when Automatic=1, else given in the data vector "canonicalNormalToCells[4,3]" where each side (up to 4) is plane with a unique fixed normal vector , $\vec{e}_d = \frac{2}{3}\vec{e}_n + \vec{e}_{sun}$ and

$F_a = -SunP.\left(\frac{AU}{R}\right)^2.abs(\vec{e}_n.\vec{e}_{Sun}).Area.(1-C_r)$ for the absorbing force,

$F_s = -2.SunP.\left(\frac{AU}{R}\right)^2.(\vec{e}_n.\vec{e}_{Sun})^2.Area.C_r.C_s$ for the specular reflection force,

$F_d = -SunP.\left(\frac{AU}{R}\right)^2.abs(\vec{e}_n.\vec{e}_{Sun}).Area.C_r.(1-C_s)$ for the diffuse reflection force,

using a "abs" function only when Automatic=1 because only in that case the two sides are subject to the Sun pressure and replacing the "abs(xx)" function by a "max (0,xx)" function for fixed sides because in that case only the side with the normal to the sun is subject to solar pressure forces (*SunP*=*SunFlux*/*c* with *c* the velocity of the light in vacuum, *SunP* ≈4.5E-6 Pa).

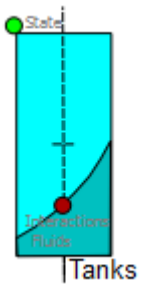
- Moment wrt O: $\vec{M}_O = \vec{OS}_{centre}\wedge\vec{F}$
- Angular momentum: $\vec{H} = \vec{0}$
- Power: $Power = \eta.SunFlux.\left(\frac{AU}{R}\right)^2.\max(0,\vec{e}_n.\vec{e}_{Sun}).Area$ using a "max" function because the solar cells are on one side only (*SunFlux*≈1353±20 W/m²) .

Note: the user can deactivate the outputs of the component by setting the user's data flag (Real) "FlagOnOff" to 0 instead of its default value 1.

13.2.11 Tanks

13.2.11.1 Description

The Tanks is a component Array of Tanks with management of the fluid interactions due to accelerations and rotations. It provides the elements to the frame component for computing the inertia matrix almost continuously. Only the n first values of the array will be used.



Note: to decrease the complexity of the inputs, the tanks are supposed to be cylindrical with flat ends. Moreover in order to decrease the complexity of the computations the tanks are discretised as a rectangular box (the square base area being adjusted for volume conservation). Thus the free surface can be for some configurations a rectangle or a square.

13.2.11.2 Construction Parameters

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Nodes	INTEGER	5	Number of nodes per axis	
n4	INTEGER	1	Number of tanks <=4	

Note to the user: be aware that the total number of node per tank is the cube of Nodes, i.e. 1000 nodes for Nodes =10. This is already a large number of nodes with Nodes =5.

13.2.11.3 PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
S	State		IN	satellite state port
if	InteractionsFluids	(n4 = n4)	OUT	for allowing the satellite to compute the Archimedes pressure for any points...

13.2.11.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
D	REAL	0.5	Diameter of every tank	m
L	REAL	1	length of every tank	m
OTankCentreLoc[4,Sat]	REAL	{ { 1,1,1} ,{ -1,-1,1} ,{...	location tank centre wrt design reference frame of the Satellite --generally launcher interface--	m
Torient[4,Sat]	REAL	{ { 0,0,1} ,{ 0,0,1} ,{ 0...	orientation axis from outlet to top (outlet is also the common bottom), not necessarily normalised	-
VolumeFluid_o[4]	REAL	{ 0.15,0.12,0 .08,0.04}	volume of fluid in each tank, rule for bi-propellants: tank1 for fuel, 2 for ox, 3 for fuel, 4 for ox	m3
rho[4]	REAL	{ 800,1400,8 00,1400}	density of the fluids; rule for bi-propellants: fuel rho[3]=rho[1] and ox rho[4]=rho[2]; Only the n first values will be used	kg/m3

13.2.11.5 DECLS:

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
Alphadeg	REAL		angle between rotation and force	deg
BondNumberWaterD	REAL		for information a Bond number at distance D from COM based on water surface tension	-
COMtank[n4,Sat]	REAL		location of COM of the fluid in each tank	m

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
			wrt design ref frame at point O	
Dsquare	REAL		individual data	m
FlagBottom[n4]	REAL		not used	-
FluidMassTanks	REAL		mass all fluids	kg
IsoDeltaPmap[n4,Nodes,Nodes,Nodes]	REAL		value of the DeltaP function for all nodes of the tank	Pa
MapFreeSurface[n4,1000,Sat]	REAL		individual data for the plot	-
MinertiaFluids[Sat,Sat]	REAL		inertia matrix all fluids wrt reference frame, not at COM	kg.m^2
MinertiaVolume[n4,Sat,Sat]	REAL		inertia matrix wrt reference frame, not at COM	kg.m^2/(kg/m^3)
Mom[Sat,Sat]	REAL		Matrix from Rotation Force frame to Sat Frame , all at the COM	-
NAccel	REAL		Norm of vector	m/s**2
NOmeg	REAL		Norm of vector	rd/s
OTankCentre[Sat]	REAL		individual data	m
OTankNodes[n4,Nodes,Nodes,Nodes,Sat]	REAL		individual data	m
OmegaX[Sat]	REAL		Rotation Force frame, temporary variables	-
OmegaY2[Sat]	REAL		Rotation Force frame, temporary variables	-
OmegaY[Sat]	REAL		Rotation Force frame, temporary variables	-
OmegaYvalid[Sat]	REAL		Rotation Force frame, temporary variables	-
OmegaZ[Sat]	REAL		Rotation Force frame, temporary variables	-
OmegaZvalid[Sat]	REAL		non Unit vectors for the Rotation Force frame, Rotation on OmegaZ , Force on eA	-
PointFreeSurfIndex[n4,3]	INTEGER		INDEXES of the location of one point of the fluid surface in each tank	-
PointFreeSurface[n4,Sat]	REAL		location of one point of the fluid surface in each tank	m
SurfTensionWater	REAL	0.07197	for information at 25°C	N/m
SurfaceIsoDeltaP[n4]	REAL		value of the DeltaP function at the surface	Pa
VolumeFluidApprox[n4]	REAL		given fluid volume in each tank and its approximation recomputed	m3
VolumeFluid[n4]	REAL		given fluid volume in each tank and its approximation recomputed	m3
VolumeTank	REAL		total volume in each tank and elementary volume	m3
XOmegaY[Sat]	REAL		Rotation Force frame, temporary variables	-
YOmegaY[Sat]	REAL		Rotation Force frame, temporary variables	-
cosAlpha	REAL		individual data	-
dPoutlet[n4]	REAL		value of the dP between outlet and surface --difference of the DeltaP function values--	Pa
dV_element	REAL		total volume in each tank and elementary volume	m3
eAX[Sat]	REAL		Unit vectors for the tank frame with axis on eAZ	-
eAY[Sat]	REAL		Unit vectors for the tank frame with axis	-

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
			on eAZ	
eAZ[Sat]	REAL		Unit vectors for the tank frame with axis on eAZ	-
eA[Sat]	REAL		Rotation Force frame, temporary variables	-
eAvalid[Sat]	REAL		non Unit vectors for the Rotation Force frame, Rotation on OmegaZ , Force on eA	-
eAxe[n4,Sat]	REAL		Unit vectors for the tank frame with axis on eAZ	-
eXsat[3]	REAL	{ 1,0,0}	reference vector	-
eYsat[3]	REAL	{ 0,1,0}	reference vector	-
nodesCube	INTEGER		total number of nodes	-
npt	INTEGER		number of points for the plot	-
sinAlpha	REAL		individual data	-

13.2.11.6 FORMULATION:

Preliminary note for the case of mono-propellant (and electric propulsion), bi-propellant tanks.

For any number of tanks the contribution of the propellant consumption to the tanks depletion is automatically performed with the following rule:

For 1 tank: the total mass-flow rate (mfr_{fuel}+mfr_{ox}) consumption from the thrusters occurs from that tank (whatever the mixture ratio of the thrusters, i.e that is like a monopropellant case)

For 2 or 4 tanks, the location of the tanks shall follow the rule:

- *The tank n°1 (and 3 if any) is stated for fuel*
- *The tank n°2 (and 4 if any) is stated for oxidizer*
- *hence the density of the fluids shall follow: $fuel_{density}$ in tanks 1 and 3 (rho[3]=rho[1]) and $oxidizer_{density}$ in tanks 2 and 4 (rho[4]=rho[2]),*
- *The mass-flow rate is divided by 2 for each tank in case of 4 tanks (equilibrium consumption)*

For 3 tanks . The tank n°1 and 3 are stated for fuel, the mass-flow rate from each of those tanks is half the total fuel propellant consumption, the tank n°2 use the total full oxidiser

The main equations to be added to the existing tank are coming from a more general Archimedes pressure. The classic relation $dp = -\rho g(z - z_0)$ is already taken into account in the tank models of ESPSS, but this is too restrictive. In space the satellite is rotating around its centre of mass (and some couples are produced by thrusters and reaction wheels, or windmill effect by the solar pressure on the solar arrays) and some non-gravity forces (from thruster, from solar pressure on the solar arrays) produce an acceleration. Thus the Archimedes pressure shall take into account the effect of the acceleration and the effect of the rotations.

The approach to model the behaviour of the liquid phase in a tank is limited to Bond Numbers B >>1 where the Bond number measures the importance of the body forces compared to the surface tension

forces $B = \frac{\rho g L}{\sigma / L}$ where g is the resultant acceleration including the rotational centrifugal force and σ the surface tension, while L is a characteristic length.

Contrary to the gravity (which induces volume forces), the thrust induces by reaction a surface force that sets the liquid side opposite to the force direction: this opposite acceleration is called inertial acceleration vector.

The effects of combined rotation and acceleration on the liquid phase location can be summarized as follow in the most general case where an instantaneous rotation $\vec{\Omega}$ and an acceleration of the satellite $\vec{A}$ are not null: one can use them for a defining a base of a frame with Z along $\vec{\Omega}$, and with $-\vec{Y}$ along $\vec{\Omega} \wedge \vec{A}$ and $\vec{X}$ along $\vec{Y} \wedge \vec{Z}$: that means that the two given vectors $\vec{\Omega}$, $\vec{A}$ are in the plane XZ. The corresponding sketch and nomenclature are given in fig.5 below.

For a point M(x,y,z) in the frame based on "$\vec{\Omega}$ ^A" the forces per unit of mass $\vec{f}$ is given by:

$$f_x = \omega^2 x + A_I \sin(\alpha)$$
$$f_y = \omega^2 y$$
$$f_z = -A_I \cos(\alpha)$$

where ω is the module of $\vec{\Omega}$ and A_I the inertial acceleration module.

The equilibrium equation of the free surface is defined by the fact that the pressure is constant.

For an infinitesimal displacement $d\vec{s} = \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix}$ we have $\vec{\nabla} P \cdot d\vec{s} = \rho \vec{f} \cdot d\vec{s}$ which give the elementary

$$dP = \rho \vec{f} \cdot d\vec{s}$$

When the small displacement belongs to the free surface, dp=0 and we finally get the equation of the free surface:

$$dP = 0 = \rho\left(f_x dx + f_y dy + f_z dz\right)$$
$$dP = 0 = \rho\left(\omega^2 x dx + A_I \sin(\alpha) dx + \omega^2 y dy - A_I \cos(\alpha) dz\right)$$

The integration of the previous equation introduces a constant:

$$0 = \rho\left(\omega^2 \frac{x^2}{2} + A_I \sin(\alpha) x + \omega^2 \frac{y^2}{2} - A_I \cos(\alpha) z\right) + Const$$

The equation of the surface can be seen as a function of x and y with the constant C_z that shall be adjusted into each tank for fulfilling the conservation of the liquid volume.

$$z = S(x,y) + C_z \quad \text{and} \quad S(x,y) = \frac{\rho}{A_I \cos(\alpha)}\left(\omega^2 \frac{x^2}{2} + A_I \sin(\alpha) x + \omega^2 \frac{y^2}{2}\right)$$

Note 1: in the exceptional case where there is absolutely no acceleration of the satellite, then the free surface is a cylinder centred on the axis of instantaneous rotation crossing the centre of mass of the satellite, it can be defined (for every z) by:

$$A_I = 0 \Rightarrow S(r) = \rho \frac{\omega^2}{2} r^2 + C_z$$

where r is the distance to the axis of instantaneous rotation and the constant C_z shall be adjusted into each tank for fulfilling the conservation of the liquid volume.

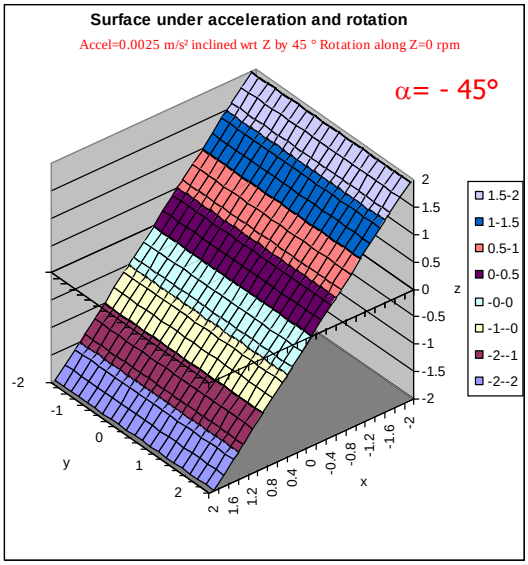
The frame based on "$\vec{\Omega}$ ^A" can be replaced by "$\vec{\Omega}$ ^eX$_{sat}$" when this product is not null (or in fine by "$\vec{\Omega}$ ^eY$_{sat}$").

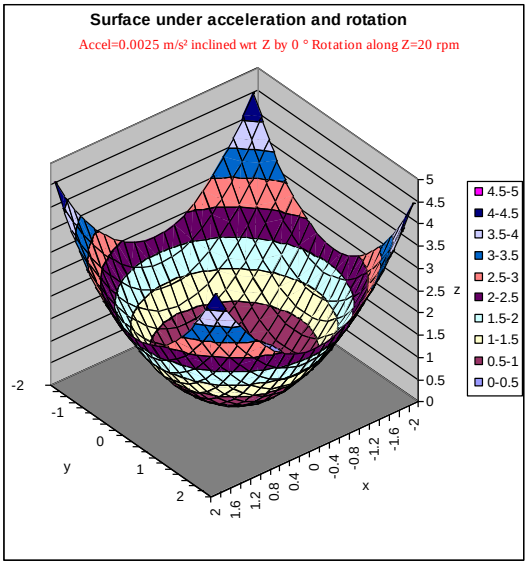
Note 2: in the exceptional case where there is absolutely no rotation of the satellite, then the free surface is a plane perpendicular to the acceleration, it can be defined (for every y) by:

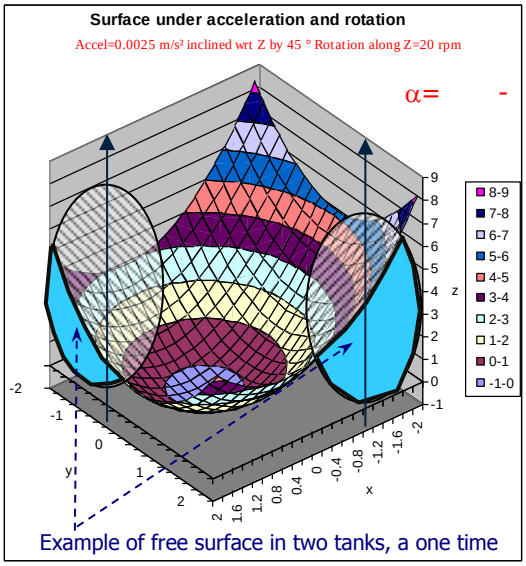
$\Delta P = 0 \Rightarrow z = S(x) = \rho.tg(\alpha)x + C_z$ where the constant C_z shall be adjusted into each tank for fulfilling the conservation of the liquid volume. This surface is a plane perpendicular to the acceleration.

Several cases of free surface are plotted in the next figures. The nomenclature is mentioned in the sketch below as well.

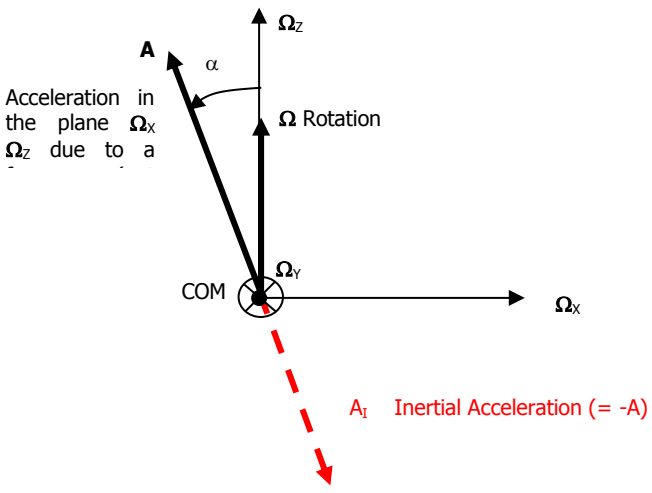
- Surface with acceleration canted at 45° and zero rotation
- Surface with acceleration and 20 rpm rotation aligned with the acceleration
- Surface with acceleration and 20 rpm rotation canted at 45° with respect to the acceleration
- Surface with rotation only
- Surface with acceleration only







Example of free surface in two tanks, a one time



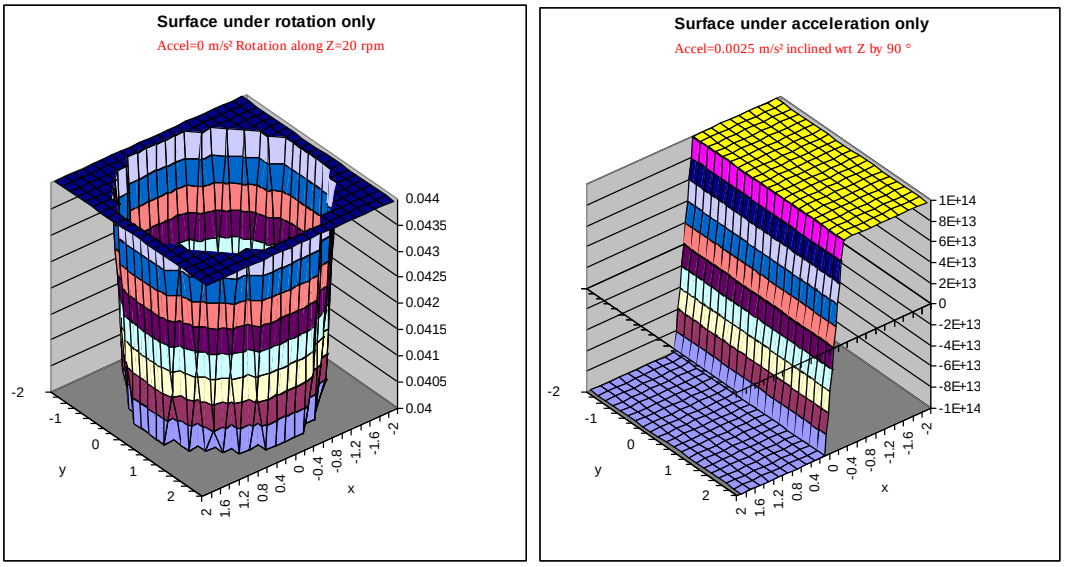


Figure 13-2: Various free surface under acceleration and rotations (see text)

13.2.11.7 Process of the computation of the general Archimedes pressure

Free Surface Method

For a given set of $\vec{\Omega}$, A and centre of mass COM, the free surface primitive can be computed in the frame deduced from $\vec{\Omega}$^A.

For each tank, the constant C_z shall be adjusted in order to satisfy to the volume conservation.
The natural frame for the tank description is the satellite frame, thus the relation of the free surface $z = S(x,y) + C_z$ in the frame deduced from $\vec{\Omega}$^A shall become a new relation $z = S_{Sat}(x,y) + C_{zSat}$ in the satellite frame thanks to a frame change which should allow computing the volume and the constant C_{zSat} for each tank.

Fluid Volume Method

For a fast resolution of the free surface, the method relies in the computation of the volume of liquid phase settled under rotation and acceleration with a systematic 3D discretisation of the tank.

With respect to the satellite frame, the tank volume is simplified with the circular base replaced by an equivalent square base:

$$Vol = \int_{Tank} dz.dx.dy$$

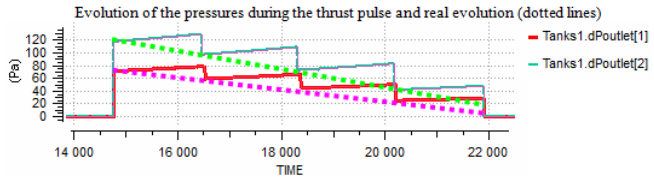
The volume of liquid phase in the tank is thus: $Vol_{LIQ} = \int_{Liquid} dz.dx.dy$

The method to select only the elements of the tank which are liquid under rotation and acceleration is based on:

- the computation of the ΔP Archimedes (in short ΔP) between each node and the COM of the satellite,
- the fact that whatever the configuration is, the liquid side will be set at the location of highest ΔP,

- and process of the free surface assessment relying on a systematic loop from the highest ΔP (most liquid side) with decreasing ΔP up to a point where the discretised volume of the liquid equal the given volume of liquid of the tank which is the free surface. In order to cover all the cases, even with liquid in both ends of the tanks, the loop systematically covers all the nodes of the tanks for a given ΔP.

  The loop process on a 3D discretisation shows that the accuracy of the results is strongly dependant on the size of the volume elements (i.e. the Nodes per axis parameter). With only 5 Nodes per axis, some slight variation of the volume may be not visible at all on the ΔP if the variation is less than the size of a layer of volume elements:
  For example, a linear decrease of the volume of fluids is transformed in this process by an evolution by steps on the ΔP, each step corresponding to a full layer of volume elements. The following plot (from ref. TN-5510 Mission Cases using EcosimPro) shows this effect in the case of a single thrust long pulse. The real evolution of ΔP with higher nodes would follow the decreasing dotted lines.



  Also from the same reference a case of 4 short thrust pulses: with 5 nodes, only half of the third thrust pulse appears with a ΔP, while with higher resolution (15 Nodes per axis) the 4 thrust pulses appears and still a step in ΔP is visible at half of the third thrust pulse.

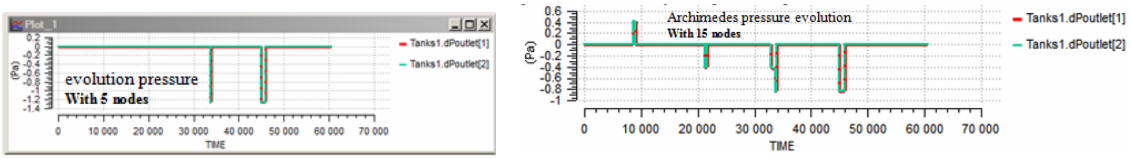


Figure 13-3: Possible effect of the number of nodes on the outputs (5 to 15 nodes per axis)

For each node element of the tank, one computes the ΔP between a node and the COM of the satellite (called iso ΔP) by:

$$iso\Delta P = \int_{COM}^{Node} dp = \int_{COM}^{Node} \rho \vec{f} \cdot d\vec{s}$$

Hence:

$$iso\Delta P = \rho\left(\omega^2\left(\frac{x^2}{2}+\frac{y^2}{2}\right)+A_I\sin(\alpha)x - A_I\cos(\alpha)z\right)_{Node} - \rho\left(\omega^2\left(\frac{x^2}{2}+\frac{y^2}{2}\right)+A_I\sin(\alpha)x - A_I\cos(\alpha)z\right)_{COM}$$

Because by definition of the frame, x,y,z are strictly null at the COM, the ΔP reduces to:

$$iso\Delta P = \rho\left(\omega^2\left(\frac{x^2}{2}+\frac{y^2}{2}\right)+A_I\sin(\alpha)x - A_I\cos(\alpha)z\right)_{Node}$$

Note: the variation of the Centre of mass in the loop process of the free surface assessment due to the real shape of the free surface in the tank could be "a priori" neglected because generally the remaining propellant mass into the tanks is quite small (about 10% or less of their initial loading mass). However, the computation of the volume allows providing the centre of mass of the liquid into each tank that can be used to update the general Satellite centre of mass.

This loop is performed in a routine "DeltaP_archimedes". The simple computation of the centre of mass of the fluid in each tank and the inertia matrix of the elements of volumes into MinertiaVolume has been added. MinertiaVolume is then multiplied by the density in each tank for getting the inertia matrixes of all the fluids wrt reference frame MinertiaFluids. A routine "InertiaMatrixDisplaced" allow setting the inertia matrix of all the fluid MinertiaFluids wrt to the COM of all the fluids. Those elements are transmitted to the frame component via the port InteractionsFluids "if".

Finally, the pressure in the liquid phase at the outlet of each tank is deduced from the free surface by the curvilinear integral along the arc from any point of the free surface to the outlet of the tank, the pressure difference is given by:

$$\Delta P_{arc} = \int_{arc} \rho \vec{f} \cdot d\vec{s} .$$

When the above iso Δ P have been computed in a discretised tank, one get simply $\Delta P_{arc} = iso\Delta P_{outlet} - iso\Delta P_{freeSurface}$ where iso Δ P$_{outlet}$ is the iso Δ P at the known node coming from the tank geometry discretisation (or any other location) and iso Δ P$_{freeSurface}$ is one node on the free surface for which the iso Δ P is the minimum in the loop on iso Δ P that provides in fine the right volume of liquid.

The above pressure difference can be positive or negative because the liquid can be in the opposite side of the exit of the tank (or any other location) while the tank's exit is still wetted by liquid thanks to the PMD inside the tank.

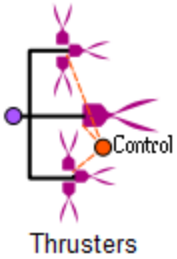
Note: The pressure would be better computed directly at the injector of each thruster: this justify the definition of the port InteractionsFluids *for transmitting all the elements needed by the dynamic function* Δ P *for enabling the computation into the connected main component (Satellite frame) for every wanted location.*

In the exceptional case where the instantaneous rotation $\vec{\Omega}$ and the acceleration are null, the inner skin of the tank is wetted uniformly by the liquid (thanks to the surface tension) and the free surface is set in the inner of the tank in which case the height with respect to the skin is computed in order to match the volume. COM of fluids and inertia matrix are computed as well in such case.

13.2.12 Thrusters

13.2.12.1 Description

The Thrusters is a component array of Thrusters of the Orbit and attitude control system. Only the n first values of the array will be used.



13.2.12.2 Construction Parameters

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
n	INTEGER	1	Number of thrusters <= 17	

13.2.12.3 PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
OnOff	PORTS_LIB.analog_signal	(n = n)	IN	Commanded OnOff full On=1 , full Off=0 (-)
f	Forces		OUT	external forces, Moment, power, flow, ... OUT port

13.2.12.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
F[17]	REAL	{ 400,10,10 ,10,10,10, 10,1...	Thrust Forces module when ON; Only the n first values will be used	N
Floc[17,3]	REAL	{ { 0,0,0} ,{ 1,1,0} ,{ 1...	location wrt design reference frame of the Satellite point O --generally launcher interface--	m
Forient[17,3]	REAL	{ { 0,0,1} ,{ 0,0,1} ,{ 0...	vector orientation not necessarily normalised	-
Isp[17]	REAL	300	Isp for each thruster; Only the n first values will be used	s
SpecificPower[17]	REAL	0	Input Power/Thrust ratio for electric propulsion for each thruster; Only the n first values will be used	W/N
rm[17]	REAL	1.65	mixture ratio = Ox/Fu as instantaneous mass flow rate for each thruster; In case of monopropellant or electric thrusters set RM=1 for a simultaneous emptying of all the tanks. Only the n first values will be used	-

13.2.12.5 DECLS:

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
ExplOnOff[n]	REAL	0	State of thruster ON = 1 (explicit)	-
Fn[17,Sat]	REAL		individual data	-
InputDataChangedFlag	REAL	1	flag for geometry input changed --into the experiment for example--	-
InputDataChangedFlag0	REAL	0	flag reference value, to be updated in discrete part	-
Mn[17,Sat]	REAL		individual data	-
OnOffChangedFlag	REAL	1	flag thrusters events	-
OnOffChangedFlag0	REAL	0	flag reference value thrusters events	-
Powern[17]	REAL		individual data	-
crossMatOFn[17,3,3]	REAL		individual data	-
eFn[17,3]	REAL		individual data	-
kk	INTEGER	0	counter	-
logEventsThrusters	STRING	""	track the thruster activity	-
mfrFuOxn[2,17]	REAL		individual data	-
mfrn[17]	REAL		individual data	-

13.2.12.6 FORMULATION:

A vector of thrusters is considered.

Each thruster is characterized by its thrust, its location at point T with respect to the geometrical frame of the satellite [O, x,y,z] (generally based on the Launcher interface, not at the gravity centre *G*, nor at the centre of mass *COM*), the orientation of the thrust $\vec{e}$, its specific impulse (*Isp*) and its specific power (*Psp*) for the case of electric thrusters.

The equations used in the components are

- Mass flow rate: $mfr = -\frac{F}{g_0 Isp}$ the negative value comes from the rule of the Ports direction and type. In case of monopropellant or electric propulsion is the mixture ratio RM shall be set to 1 if the number of tanks is >1 in order to get simultaneous emptying.
  For the oxidiser: mfr[2]= $mfr_{ox} = mfr\frac{RM}{RM+1}$ and for the fuel mfr[1]= $mfr_{fu} = mfr - mfr_{ox}$
- Force: $\vec{F} = F.\vec{e}$ where $\vec{e}$ is the unit vector of the thrust direction
- Moment wrt O: $\vec{M}_O = \vec{OT} \wedge \vec{F}$ (swirl to be considered in a further phase)
- Angular momentum: $\vec{H} = \vec{0}$
- Power: $Power = -Psp.F$ the negative value comes from the rule of the Ports direction and type.

  Note: the input signal "OnOff.signal[n]" is saved into an explicit variable "ExplOnOff[n]" that is used further in the continuous part for enabling or not the component equations outputs values.

The thruster activity is tracked into a compact string variable "logEventsThrusters": for each event of switching on or off one or several thruster, the variable logEventsThrusters is changed with the numbering of the thruster that are active : example "7 9" when the thruster 7 and 9 are switched on. This is performed in the discrete part of the component thanks to a flag "OnOffChangedFlag" watching in the continuous part any changes in the thruster activity.

13.3 FUNCTIONS OF THE SATELLITE LIBRARY

Several basic functions have been defined, their definition is standard:

- rad (convert degrees into radian); degfull (convert radian in degrees without limit, used for rotation speeds); deg (convert radian in degrees within 0, 360°); SkewSymmetricLowTriangle (to fill the low triangle of a triangular matrix), CrossMatrix and CrossMatrixEnum (matrix form for performing more easily a cross product of vectors); XcrossY; XscalarY; Norm; Normalise (normalise a vector); aXb (linear combination of a vector X); aXbYc; Integer; ErrRounding (manage some errors due to roundings);

Quaternion functions

- CheckUnitQ (for unit quaternions with renormalisation in order to reduce the rounding errors); CheckChoiceQuater (return the unit quaternion with positive real component); QMatrix; QMatrixHat (for enabling the quaternion products), Q1productQ2; Qinverse;

Function Coordinate frame change and Quaternions:

Preliminary Note: the naming convention for Frame change functions is highlighted
Naming convention **CoordA~~~~~~B#######** for a given vector in Frame A to be converted in Frame B coordinates. Moreover **~~~~~~** designate the METHOD that from the frame A orients the frame B: rotation, Euler angle, Cardan angles, quaternion, matrix.
When the METHOD that orients the frames is reversed then this is added with underscore like: toB_BrotationA. In **#######** the output of the function is given: Euler angle, Cardan angles, quaternion, matrix or nothing for a vector output
In short we have: CoordA METHOD B OUTPUT or CoordAtoB B METHOD A OUTPUT

- CoordAtoB_BrotationA; CoordArotationB; CoordAtoB_BeulerA;
- CoordAqaterBmatrix; CoordAqaterBmatrixEnum; CoordAcardanBmatrix; CoordAqaterBeuler; CoordAquaterBcardan; CoordOrbCardanSatQuater; CoordECIeulerOrbQuater;

Functions Specialised for orbit:

- JulianDay; CoordPolEulerECI; CoordRVECItoEulerAngles;
- MoonSunECI; ParametersToCartesian; Eclipse(set a flag when in eclipse)

Functions Specialised for attitude:

- DervativeQmatrixSatAxis; InertiaMatrixSatAxis; InertiaMatrixDisplaced; InertiaMatrixTotalDisplaced; InertiaMatrixLoop

Functions Specialised for components:

- CrossMatrixIndex; MatrixFrameChange; AngleVectordeg; FrameChange; deltaP; ComputeDeltaPmap; MinMaxIsoDeltaPmap; DeltaP_archimedes; EventsThrusters(set a compact string with thrusters ON)

The list of functions in alphabetic order is the following:

- AngleVectordeg "gives the angle from vector A to vector B"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
A[Sat]	IN REAL				
B[Sat]	IN REAL				
cosAlpha	OUT REAL				
sinAlpha	OUT REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Alphadeg	REAL				
C[Sat]	REAL				
aa	REAL				
bb	REAL				
sinAlpha_abs	REAL				

- CheckChoiceQuater "Make Unit and perform the choice of the quaternion with a positive real part because equivalent to the one with the angle (2PI-angle)/2 and axis -U"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Q[4]	OUT REAL				
Qo[4]	IN REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
NormQ	REAL				

- CheckUnitQ "to check the Quaternion unit norm and make it unit again; to be used only in Function because OUT but cannot be used in Continuous part"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Message	IN STRING	""			
Q[4]	OUT REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CheckUnit	REAL				
NormQ	REAL				
ii	INTEGER				

- ComputeDeltaPmap "Function that return the value of the dP fulfilling the general Archimedes pressure equation under rotation and acceleration for a given map of points in the Omega and F frame"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
IsoDeltaPmap[n4,Nodes,Nodes,Nodes]	OUT REAL			iso dP values wrt COM of the Sat	Pa
Mom[Sat,Sat]	IN REAL			Matrix based on Omega	-
NAccel	IN REAL			parameter for the function deltaP()	m/s2
NOmeg	IN REAL			parameter for the function deltaP()	rd/s
Nodes	IN INTEGER			number of nodes per axis	-
OCOM[Sat]	IN REAL			Location of the COM with respect to the origin O: vector O to C	m
OTankNodes[n4,Nodes,Nodes,Nodes,Sat]	IN REAL			All the nodes location (fixed in Sat)	m
cosAlpha	IN REAL			parameter for the function deltaP()	-
n4	IN INTEGER			number of tanks	-
rho[n4]	IN REAL			parameter for the function deltaP()	kg/m3
sinAlpha	IN REAL			parameter for the function deltaP()	-

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
NodeOmega[Sat]	REAL				

- ComputeDeltaPpoint: "Function that return the value of the dP fulfilling the general Archimedes pressure equation under rotation and acceleration for a given map of points in the Omega and F frame"

RETURN TYPE: REAL

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Mom[Sat,Sat]	IN REAL			Matrix based on Omega	-
NAccel	IN REAL			parameter for the function	m/s2

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
				deltaP()	
NOmeg	IN REAL			parameter for the function deltaP()	rd/s
OCOM[Sat]	IN REAL			Location of the COM with respect to the origin O: vector O to C	m
OPoint[Sat]	IN REAL			Location of a point P with respect to the origin O: vector O to P	m
cosAlpha	IN REAL			parameter for the function deltaP()	-
rho	IN REAL			parameter for the function deltaP()	kg/m3
sinAlpha	IN REAL			parameter for the function deltaP()	-

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
DeltaPpoint	REAL				
PointOmega[Sat]	REAL				

- CoordAcardanBmatrix "Frame change matrix to the one oriented by the Cardan Vector[B] = MatrixAtoB * Vector[A]"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
MatrixAtoB[3,3]	OUT REAL				
phi	IN REAL			Roll angle (rad)	rad
psi	IN REAL			Yaw angle (rad)	rad
theta	IN REAL			Pitch angle (rad)	rad

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
cosph	REAL				
cosps	REAL				
cost	REAL				
sinph	REAL				
sinps	REAL				
sint	REAL				

- CoordAqaterBeuler "Compute the Euler's angles that orient FrameA to FrameB from the Quaternion "

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
OM	OUT REAL				
Q[4]	IN REAL				
inc	OUT REAL				

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
omPHI	OUT REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CheckAcos	REAL				

- CoordAqaterBmatrix "Frame change matrix to the one oriented by the Quaternion Vector[B] = MatrixAtoB * Vector[A]"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
MatrixAtoB[3,3]	OUT REAL				
Message	IN STRING	""			
Q[4]	IN REAL			Quaternions coordinates wrt frame A , real in Q[1] (-)	-

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
QMX[4,4]	REAL				
QMXhat[4,4]	REAL				
Qinv[4]	REAL				

- CoordAqaterBmatrixEnum "Frame change matrix to the one oriented by the Quaternion Vector[B] = MatrixAtoB * Vector[A]"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
MatrixEnum[]	OUT REAL				
Q[4]	IN REAL			Quaternions coordinates wrt frame A , real in Q[1] (-)	-

- CoordAquaterBcardan "Compute the Cardan angles psi, theta, phi on yaw, pitch, roll around Z, Y', x that orient FrameA to FrameB from Quaternion "

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Q[4]	IN REAL			Quaternions coordinates wrt frame A with real in Q[1] (-)	-
phi	OUT REAL			Roll angle (rad)	rad
psi	OUT REAL			Yaw angle (rad)	rad
theta	OUT REAL			Pitch angle (rad)	rad

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CheckAsin	REAL				

- CoordArotationB "Frame change: vector in frameA to same vector in frameB where Frame A after rotation gives Frame B"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Anglerad	IN REAL				
Xa	IN REAL				
Ya	IN REAL				
xb	OUT REAL				
yb	OUT REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CosAngle	REAL				
SinAngle	REAL				

- CoordAtoB_BeulerA "Frame change: vector in frameA to same vector in frameB where Frame B after Euler's --PrecessionZB, NutationU, RotationZA-- gives Frame A "

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CoordInA[]	IN REAL				
CoordInB[]	OUT REAL				
NutationU	IN REAL				
PrecessionZB	IN REAL				
RotationZA	IN REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Tau	REAL				
Uu	REAL				
vv	REAL				

- CoordAtoB_BrotationA "Frame change: vector in frameA to same vector in frameB where Frame B after rotation gives Frame A"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Anglerad	IN REAL				
Xa	IN REAL				
Ya	IN REAL				
xb	OUT REAL				
yb	OUT REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CosAngle	REAL				
SinAngle	REAL				

- CoordECIeulerOrbQuater" Compute the Quaternion that orient from the ECI to Orb with Euler's angles given from ECI to Pol. Please note that the quaternion is ECI to Orb and not to Pol

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
OM	IN REAL				
Q[4]	OUT REAL				
inc	IN REAL				
omPHI	IN REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Q1[4]	REAL				
Q2[4]	REAL				
Q3[4]	REAL				
Q4[4]	REAL				
Q5[4]	REAL				
QQQQ[4]	REAL				
QQQ[4]	REAL				
QQ[4]	REAL				

- CoordOrbCardanSatQuater "Compute the Quaternion that orient Orb to Sat, from the Cardan angles psi, theta, phi on yaw, pitch, roll around Z, Y', x"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Q[4]	OUT REAL			Quaternions coordinates wrt frame A with real in Q[1] (-)	-
phi	IN REAL			Roll angle (rad)	rad
psi	IN REAL			Yaw angle (rad)	rad
theta	IN REAL			Pitch angle (rad)	rad

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Q1[4]	REAL				
Q2[4]	REAL				
Q3[4]	REAL				
QQ[4]	REAL				

- CoordPolEulerECI " ECI after Precession, Nutation, Rotation gives Pol Axis --this is not Orb for telecom satellites-- "

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CoordECI[]	OUT REAL				
CoordOrbitalPolar[]	IN REAL				
Nutation	IN REAL				
Precession	IN REAL				
Rotation	IN REAL				

- CoordRVECItoEulerAngles " Compute the Euler angles that orient FrameECI to FrameORB from orbital R V in ECI and several Keplerian parameters (osculating) of the orbit"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Alt	OUT REAL				
AltApo	OUT REAL				
AltPer	OUT REAL				
Exc	OUT REAL				
GMfocus	IN REAL				
Phi	OUT REAL				
RAAN	OUT REAL				
R[ECI]	IN REAL				
Rfocus	IN REAL				
V[ECI]	IN REAL				
inc	OUT REAL				
om	OUT REAL				
omPHI	OUT REAL				
sma	OUT REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CheckNorm	REAL				
Checkasin	REAL				
CosE	REAL				
CosOm	REAL				
CoshH	REAL				
Cosi	REAL				
CosomPhi	REAL				
Energy	REAL				
HH	REAL				
H[ECI]	REAL				
SinE	REAL				
SinOm	REAL				
SinhH	REAL				
Sini	REAL				
SinomPhi	REAL				
cosphi	REAL				
dtp	REAL				
e	REAL				
eH[ECI]	REAL				
eR[ECI]	REAL				

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
eV[ECI]	REAL				
eX[ECI]	REAL				
eY[ECI]	REAL				
eZ[ECI]	REAL				
hh	REAL				
mR[ECI]	REAL				
mean	REAL				
n	REAL				
rr	REAL				
sinphi	REAL				
vv	REAL				

- CrossMatrix "Cross matrix for easily compute a vector cross product XxAnyOtherVector"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CMX[3,3]	OUT REAL				
X[]	IN REAL				

- CrossMatrixEnum " cross product of vectors with Enum type "

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CMX[]	OUT REAL				
V[]	IN REAL				

- CrossMatrixIndex " cross product matrix dedicated for 3 dimensions vector time n17 "

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Index	IN INTEGER				
iCMV[n17,3,3]	OUT REAL				
iV[n17,3]	IN REAL				
n17	IN INTEGER				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CMV[3,3]	REAL				
V[3]	REAL				

- DeltaP_archimedes "Function that return the DeltaP_archimedes and more. Discretisation in steps between min and Max : as the general one =Nodes"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
COMtank[n4,Sat]	OUT REAL			individual COM in Sat	m
FlagBottom[n4]	IN REAL			not used	-

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
IsoDeltaPmap[n4, Nodes,Nodes,Nodes]	IN REAL			Iso dP values wrt COM of the Sat for all nodes	Pa
MapFreeSurface[n4,1000,Sat]	OUT REAL			for plot of the surface for each tanks	m
MinertiaVolume[n4,Sat,Sat]	OUT REAL			inertia/rho of each fluid in the tank, wrt Sat	kg.m2/(kg/m3)
Mom[Sat,Sat]	IN REAL			Matrix based on Omega	-
Nodes	IN INTEGER			number of nodes per axis	-
OTankNodes[n4, Nodes,Nodes,Nodes,Sat]	IN REAL			All the nodes location (fixed in Sat)	m
PointFreeSurfIndex[n4,3]	OUT INTEGER			index	-
PointFreeSurface[n4,Sat]	OUT REAL			location of one point on the surface for each tanks	m
SurfaceIsoDeltaP[n4]	OUT REAL			iso dP of any point on the surface for each tanks	Pa
VolumeFluidApprox[n4]	OUT REAL			fluid volume computed for each tanks	m3
VolumeFluid[n4]	IN REAL			volume of tanks	m3
dPoutlet[n4]	OUT REAL			dP = iso dP outlet - iso dP surface for each tanks	Pa
dV_element	IN REAL			volume of every node	m3
n4	IN INTEGER			number of tanks	-
npt	OUT INTEGER			number of points used for the plot of the surface	-

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
ConditionOnlyLastLayer	BOOLEAN	FALSE			
IsoDeltaP	REAL				
IsoDeltaP_outlet	REAL				
IsoDeltaPbottom	REAL				
IsoDeltaPmax[n4]	REAL				
IsoDeltaPmin[n4]	REAL				
NodesdeltaP	INTEGER				
SurfaceIsoDeltaPmin	REAL				
Vol	REAL				
Volpreviouslayer	REAL				
VolumeFluidApproxRough	REAL				
VolumeWall	REAL				
VolumeWallLoop	REAL				
dP	REAL				
dV_elementAdjusted	REAL				
deltaPM	REAL				
deltaPm	REAL				

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
flag	INTEGER				
iCentre	INTEGER				
iHalfNodes	INTEGER				
iOutlet	INTEGER				
iSurface	INTEGER				
inb	INTEGER				
inbNodesLayer	INTEGER				
inum	INTEGER				
nptplus	INTEGER				
step	REAL				

- DervativeQmatrixSatAxis "gives the Derivative matrix for quaternions"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
DQmat[4,4]	OUT REAL				
Wrot[Sat]	IN REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
QWrot[4]	REAL				
ii	INTEGER	1			

- Eclipse" Function that return the total eclipse flag "

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
FlagEclipse	OUT BOOLEAN				
FractionSun	OUT REAL				
RSatEarth[Sat]	IN REAL				
RSatSun[Sat]	IN REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Hatmosphere	REAL	100000 * 0		no penumbra, h=0	m
RConeEarth[Sat]	REAL				m
RConeSat[Sat]	REAL				m
Rearth	REAL	6378137			m
RearthSun[Sat]	REAL				m
Rsun	REAL	109 * 6378137			m
angleAtConed	REAL	0			°
angleAtConedTot	REAL			not used, for penumbra and total eclipse	°
angled	REAL	0			°
halfconed	REAL	0			°
halfconedLimit	REAL			not used, for penumbra and total	°

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
				eclipse	
halfconedTot	REAL			not used, for penumbra and total eclipse	°

- ErrRounding" Message error routine"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Message	IN STRING				
RoundingErrorAcceptable	IN REAL				
Small	IN REAL				

- EventsThrusters "produce a simple compact log of active devices Thrusters1.logEventsThrusters= 7 9"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
n	IN INTEGER				
signal[n]	IN REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Event	STRING	""			
ii	REAL				

- FrameChange "from a vector in SatAxis --origin O-- produces the vector in the Omega and F frame starting from current COM"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Mom[Sat,Sat]	IN REAL				
NodeOmega[Sat]	OUT REAL				
NodeSatAxis[Sat]	IN REAL				
OCOM[Sat]	IN REAL				

- InertiaMatrixDisplaced " inertia matrix displaced from its own COM to other point; with a flag set to FALSE if it is wanted from a point to its own COM (at its own COM the matrix is at minimum amplitude)"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
FromCOMNotToCOM	IN BOOLEAN	TRUE		when false, the new reference shall be necessarily its own COM	
InertiaMatrixD[Sat,Sat]	OUT REAL			InertiaMatrix displaced by Rd to the new reference	
InertiaMatrix[Sat,Sat]	IN REAL			InertiaMatrix wrt reference (by default it is wrt its own	by default it is wrt its

				COM)	own COM
Mass	IN REAL			total mass of involved in the object characterized by the given InertiaMatrix	
Rdisplaced[Sat]	IN REAL			vector from the InertiaMatrix reference to the point wanted (or necessarily to its own COM when FromCOMNotToCOM=FALSE)	or necessarily to its own COM when

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Idelta[Sat,Sat]	REAL				
RR2	REAL				
RRIdent3[Sat,Sat]	REAL				
RRdisplaced[Sat]	REAL				

Id[i,j] =I[i,j]+ Mass*(Kroneker[i,j]*Rd**2-Rd[i]*Rd[j]) where I must be the Inertia matrix wrt to the COM of object characterized by I and Rd must be the vector from the object's COM to other point, Id is the matrix displaced (for the default case flag FromCOMNotToCOM =TRUE). The equation above is reversed for the case flag FromCOMNotToCOM =FALSE

- InertiaMatrixLoop" for a loop on the indexes l ,m, n compute the inertia/rho increments (i.e. by volume) and add to the matrix"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
MinertiaVolume[n4,Sat,Sat]	OUT REAL				
Nodes	IN INTEGER				
OTankNodes[n4,Nodes,Nodes,Nodes,Sat]	IN REAL				
dV_element	IN REAL				
i	IN INTEGER				
l	IN INTEGER				
m	IN INTEGER				
n	IN INTEGER				
n4	IN INTEGER				

- InertiaMatrixSatAxis2 "Set the inertia matrix from the inputs"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
InertiaMatrixSatFrozen[Sat,Sat]	OUT REAL				
Ixxyyzz[Sat]	IN REAL				
Ixyyzzx[Sat]	IN REAL				

- InertiaMatrixTotalDisplaced" inertia matrix non-mobile DRY and FLUIDS displaced from their own COM to the current COM of the sat"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
FluidMassTanks	IN REAL				
InertiaMatrixFluidsCfluids[Sat,Sat]	IN REAL				
InertiaMatrixSatFrozenComdry[Sat,Sat]	IN REAL				
InertiaMatrixSatFrozen[Sat,Sat]	OUT REAL				
OCOMFluids[Sat]	IN REAL				
OCOM[Sat]	IN REAL				
OCOMdry[Sat]	IN REAL				
massDry	IN REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CdryCOM[Sat]	REAL				
CfluidCOM[Sat]	REAL				
InertiaMatrixFluids[Sat,Sat]	REAL				
InertiaMatrixSatFrozenDry[Sat,Sat]	REAL				

- Integer "Return an integer value, toward zero when negative"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Real	IN REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
i	INTEGER				

- JulianDay "compute the Julian day from a Gregorian Calendar date in string; JD Julian day counting starting from 0 in -4712 i.e. 4713BC; function ok for dates > 1500"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
DDMMYYYYHHMMSS	IN STRING				
Day1900	OUT REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
BC	INTEGER	0		Before Christ	
Day	INTEGER				
Hour	INTEGER				
JD	REAL	0			
JD1900	REAL	2415020.5			
Minute	INTEGER				

Month	INTEGER				
Second	INTEGER				
Year	INTEGER				
a	INTEGER	0		Before Christ	
m	INTEGER	0		Before Christ	
v1	REAL				
v2	REAL				
y	INTEGER	0		Before Christ	

- MatrixFrameChange "set a matrix for frame change defined by 3 vectors of base"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Mom[Sat,Sat]	OUT REAL				
OmegaXo[Sat]	IN REAL				
OmegaYo[Sat]	IN REAL				
OmegaZo[Sat]	IN REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
OmegaX[Sat]	REAL				
OmegaY[Sat]	REAL				
OmegaZ[Sat]	REAL				

- MinMaxIsoDeltaPmap "Function that return the MinMax value of the dP from IsoDeltaPmap"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
IsoDeltaPmap[n4,Nodes,Nodes,Nodes]	IN REAL				
IsoDeltaPmax[n4]	OUT REAL				
IsoDeltaPmin[n4]	OUT REAL				
Nodes	IN INTEGER				
n4	IN INTEGER				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
NodeOmega[Sat]	REAL				

- MoonSunECI "produce the location of Moon and Sun in ECI equatorial for dates in 2000 +-100 years"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
JD	IN REAL				
ReMo[ECI]	OUT REAL				
ReS[ECI]	OUT REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
A0[25]	REAL				
AU	REAL	149597870000.			
B0[25]	REAL				
Bs	REAL				
C0[25]	REAL				
CosAngle	REAL				
DiamKm[25]	REAL				
Diff	REAL				
F	REAL				
FoyerSatellites[25]	REAL				
IterationKeplersp[25]	REAL				
JDbouiges	REAL				
L0[25]	REAL				
LongOrb	REAL				
LorbitalSun	REAL				
Lp[25]	REAL				
Ltilda	REAL				
LtildaVraie	REAL				
M	REAL				
M0[25]	REAL				
MorbitalSun	REAL				
Ms	REAL				
OM	REAL				
OM0[17]	REAL				
OMp[17]	REAL				
PI	REAL	3.141592653589			
ReMoEcliptic[3]	REAL				
ReSEcliptic[3]	REAL				
Rs	REAL				
RsEEcliptic[3]	REAL				
RsMoEcliptic[3]	REAL				
SinAngle	REAL				
aMajorsp[25]	REAL				
b	REAL				
exc[17]	REAL				
i	INTEGER				
incl[17]	REAL				
l	REAL				
lh	REAL				
om0[17]	REAL				
omB	REAL				
omP[17]	REAL				
parallaxe	REAL				
u	REAL				
v	REAL				

- Norm "norm of X; case 0 managed"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
X[]	IN REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
X_module2	REAL				

- Normalise " output a unit vector eX from X; except when null "

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Message	IN STRING	""			
X[]	IN REAL				
eX[]	OUT REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
X_module2	REAL	0.7			
XsumAbs	REAL	1			
checknorm	REAL	0.7			
eX_module2	REAL	0.7			

- NormaliseIndex " Normalise an array of vectors"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Index	IN INTEGER				
Message	IN STRING				
X[n,3]	IN REAL				
eX[n,3]	OUT REAL				
n	IN INTEGER				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
V[3]	REAL				
eV[3]	REAL				

- ParametersToCartesian "give the radius and velocity of an orbit defined by its orbital parameters"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
OM_o	IN REAL			RAAN --also called OMEGA-- of the initial orbit	deg
PHI_o	IN REAL			true anomaly	deg
R_o[ECI]	OUT REAL				m
V_o[ECI]	OUT REAL				m/s

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
altApo_o	IN REAL			Perigee altitude from Earth	m
altPer_o	IN REAL			Apogee altitude from Earth	m
error[4]	OUT REAL			rounding errors, summary in index 1: error[1]	-
inc_o	IN REAL			Inclination of the initial orbit	deg
om_o	IN REAL			perigee argument --also called omega-- of the initial orbit	deg

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
C_orbit	REAL				
EarthR	REAL	6378137		equatorial earth radius	m
GMEarth	REAL	398600500000000.		G.M_earth	m3/s2
LocalOrbitalPolar[Pol]	REAL				
Rrr[3]	REAL				
V_module	REAL				
Vvv[3]	REAL				
Wenergy	REAL				
checkr	REAL				
checkv	REAL				
checkvv	REAL				
exc	REAL				
p_orbit	REAL				
r_module	REAL				
sma	REAL				
theta_o	REAL				
v_radial	REAL				
v_theta	REAL				

- Q1productQ2 " In quaternion algebra Q1={r,v} Q1={r',v'} with r real part, v pure quaternion like a vector: quaternion product= {rr'-v.v',rv'+r'v+v x v'} "

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Message	IN STRING	""			
Q1Q2[4]	OUT REAL				
Q1[4]	IN REAL				
Q2[4]	IN REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CheckUnit	REAL				
Error	REAL	0			
QMX[4,4]	REAL				
X[3]	REAL				
XxY[3]	REAL				
Y[3]	REAL				
ii	INTEGER				

- QMatrix: "Quaternion matrix --skew-symmetric matrix-- for easily compute a product Q.AnyOtherQuaternion = [QMX].AnyOtherQuaternion in matrix »

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
QMX[4,4]	OUT REAL				
Q[4]	IN REAL				

  In quaternion algebra Q={r,v} Q'={r',v'} with r real part, v pure quaternion like a vector Quaternion product= {rr'-v.v',rv'+r'v+v x v'} can be set directly in form of a matrix Q.Q' =[QMX].Q'
  This function set the diagonal and upper triangle column>row for [row, column] matrix, then a function is called for filling the lower triangle.

- QMatrixHat: "Quaternion matrix --skew-symmetric matrix-- for easily compute a product AnyOtherQuaternion.Q = [QMXhat].AnyOtherQuaternion in matrix. Note that here the matrix refer to AnyOtherQuaternion.Q and not to Q.AnyOtherQuaternion "

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
QMXhat[4,4]	OUT REAL				
Q[4]	IN REAL				

This corresponds to the commutation of the product because generally not commutative.

- Qinverse: In quaternion algebra for unit quaternion, inverse = conjugate i.e. Qinverse=Q* "

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
Q[4]	IN REAL				
Qconjug[4]	OUT REAL				

- SkewSymmetricLowTriangle: "once upper triangle column>row for mat[row, column] are given, this function Skew Symmetric the Lower Triangle row>column "

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
mat[n,n]	OUT REAL				
n	IN INTEGER				

- XcrossY "cross product of vectors XxY using Cross matrix for vectorX "

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
X[]	IN REAL				
XxY[]	OUT REAL				
Y[]	IN REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
CMX[3,3]	REAL				

- XscalarY "Scalar product of vectors X.Y"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
X[]	IN REAL				
Y[]	IN REAL				

- aXb "linear combination to give a.VectorX+b"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
X[]	IN REAL				
a	IN REAL				
aXplusb[]	OUT REAL				
b	IN REAL				

- aXbYc "linear combination to give a.VectorX+b.VectorY+c"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
X[]	IN REAL				
Y[]	IN REAL				
a	IN REAL				
aXplusbYplusc[]	OUT REAL				
b	IN REAL				
c	IN REAL				

- deg "set radian to degrees within [0,360["

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
radians	IN REAL				

  DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
deg_full	REAL				
deg_modulo	INTEGER				

- degfull "set radian to degrees without limits in [0,360[, used for the speed °/s"

  ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
radians	IN REAL				

- deltaP "Function that return the value of the constant dP --units: pressure Pa-- for fulfilling the general Archimedes pressure for a given point in the Omega and F frame with origin at the COM"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
NAccel	IN REAL				
NOmeg	IN REAL				
NodeOmega[Sat]	IN REAL				
cosAlpha	IN REAL				
rho	IN REAL				
sinAlpha	IN REAL				

DECLS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
dP	REAL				

- rad "set degrees to radians"

ARGUMENTS:

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
degrees	IN REAL				

13.4 EXAMPLE OF USE

A stand-alone example of every single component is barely meaningful. Instead a North-South Electric propulsion manoeuvre is presented for the demonstration of the use of the orbital characteristics of the Satellite library

Model: A satellite attitude Earth pointing and with 2 thrusters used for performing the North-South correction.

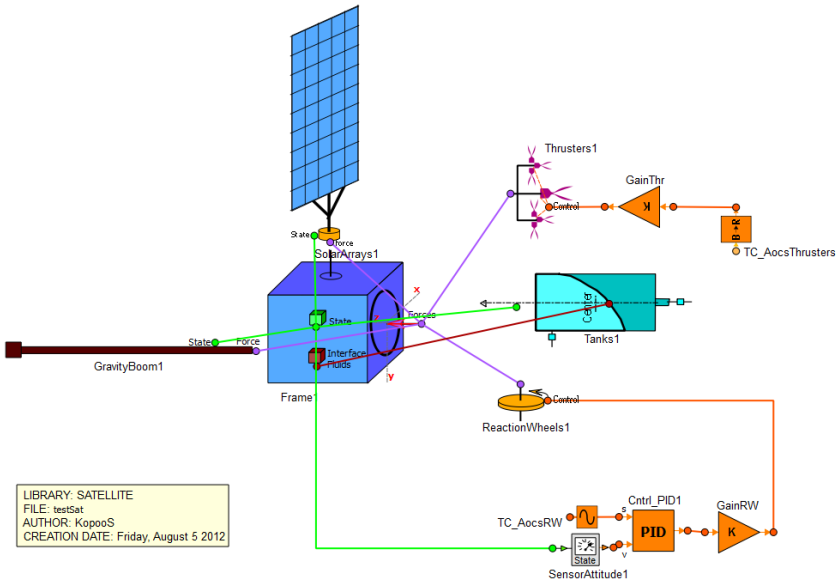


Figure 13-4: Model for North-South station keeping with Satellite Library

The reference case is given by a freeware "TriaXOrbitaL" available on the web. Thus the initial mass, the thruster performances and the timing of the manoeuvres is provided by this reference. The initial orbit with the trajectory followed by the satellite after the thrust periods is shown in the output plot of the reference (J2, Moon-Sun perturbations deactivated)

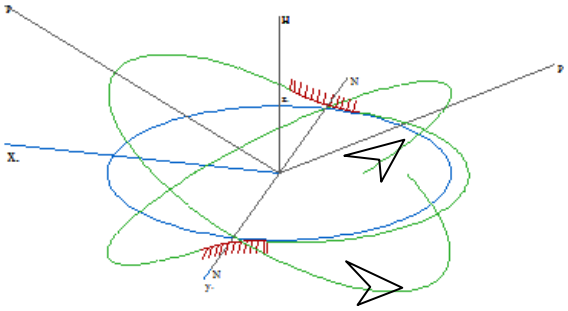


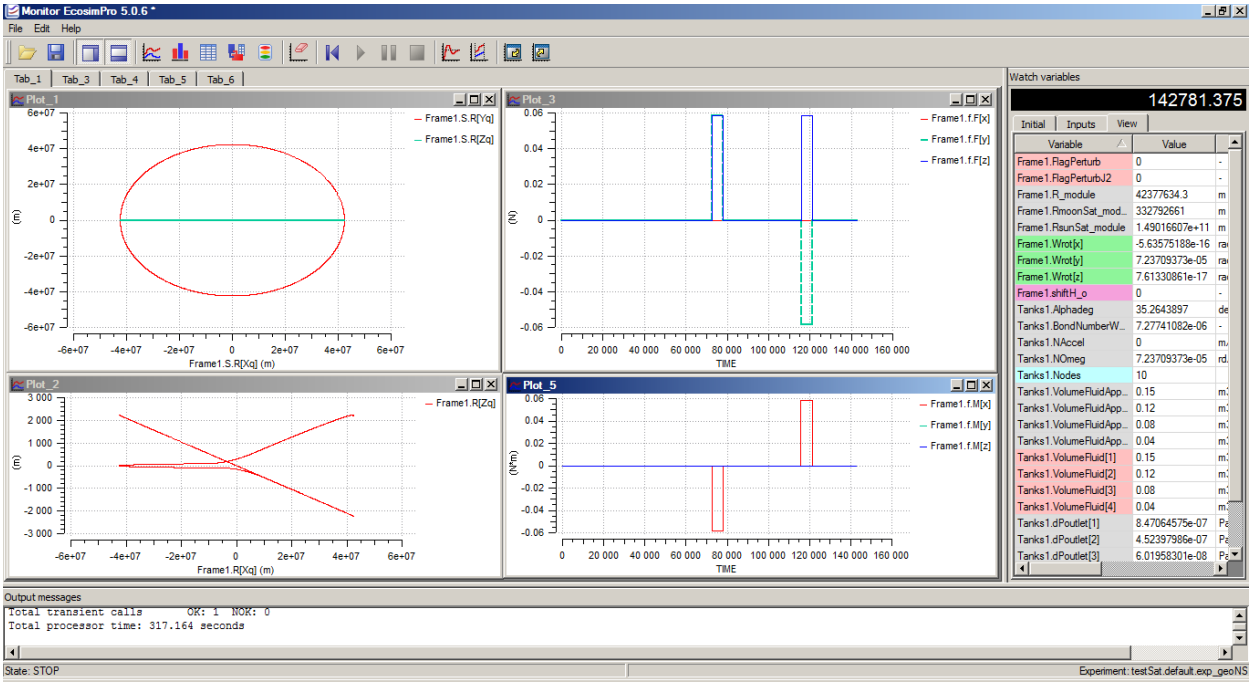
Figure 13-5: North-South station keeping with Reference Tool (altitudes with respect to GEO +-20 km with zoom amplification in inclination)

The Experiment under EcosimPro is

```
        -- ' 30/08/2012   22:37:26
EXPERIMENT exp_geoNS ON testSat.default
DECLS
        STRING filename="Rep"
        REAL Fthrust=1
        REAL dt=1
        INTEGER t
INIT
        -- initial values for state variables
        Cntrl_PID1.vi[1] = 0
        Cntrl_PID1.yf[1] = 0
        Cntrl_PID1.vi[2] = 0
        Cntrl_PID1.yf[2] = 0
        Cntrl_PID1.vi[3] = 0
        Cntrl_PID1.yf[3] = 0
BOUNDS
        -- Set equations for boundaries: boundVar = f(TIME;...)
        ReactionWheels1.rpmControl[4] = 0
BODY
        RDIGITS=16
        ABS_ERROR=1E-7 --10
        REL_ERROR=ABS_ERROR
        Frame1.DDMMYYYYHHMMSS="21032015000000"
        --Orbit GEO
        Frame1.altApo_o=36000001.
        Frame1.altPer_o=36000000.
        Frame1.RAAN_o=90 --90.
        Frame1.inc_o=0.003
        Frame1.om_o=0
        Frame1.PHI_o=45
        Frame1.massDry_o = 2000.
        --Model Perturbations
        Frame1.FlagPerturb=0 --NO Sun Moon
        Frame1.FlagPerturbJ2=0 -- NO J2
        SolarArrays1.FlagOnOff=0 --NO Sun Pressure
        GravityBoom1.FlagOnOff=0 -- NO GravityBoom
        --AOCS settings Earth pointing and perturbations with some Thrusters TC
        TC_AocsRW.Amp=0.
        Fthrust=0.083/10 -- adjustment thrust level to 83 mN for a default value of 10 N
```

```
FOR (i IN 1,13)
    GainThr.k[i]=Fthrust*GainThr.k[i]
END FOR
  Thrusters1.Isp=1600
  TIME = 0 -- set before using AFTER statement
  dt=72631.5
  t=9 --loc { 0,-1,0} orient { 0,1,1} force South
  TC_AocsThrusters.s_in.signal[t]=TRUE AFTER dt
  TC_AocsThrusters.s_in.signal[t]=FALSE AFTER dt+2762.1*2
  dt=115716.3
  t=8 --loc { 0,-1,0} orient { 0,1,1} force North
  TC_AocsThrusters.s_in.signal[t]=TRUE AFTER dt
  TC_AocsThrusters.s_in.signal[t]=FALSE AFTER dt+2762.1*2
  --Loop settings
  SensorAttitude1.Gain=1 --10
  ReactionWheels1.tau=0.01
  GainRW.k[1]=10000
  FOR (i IN 2,3)
    GainRW.k[i]=GainRW.k[1]
  END FOR
  --Integration
  TSTOP =121240.4+86163.9/4 --86400
  CINT = TSTOP/10
  INTEG()
END EXPERIMENT
```

The results on the Monitor EcosimPro are comparable to the reference according to the following plot.



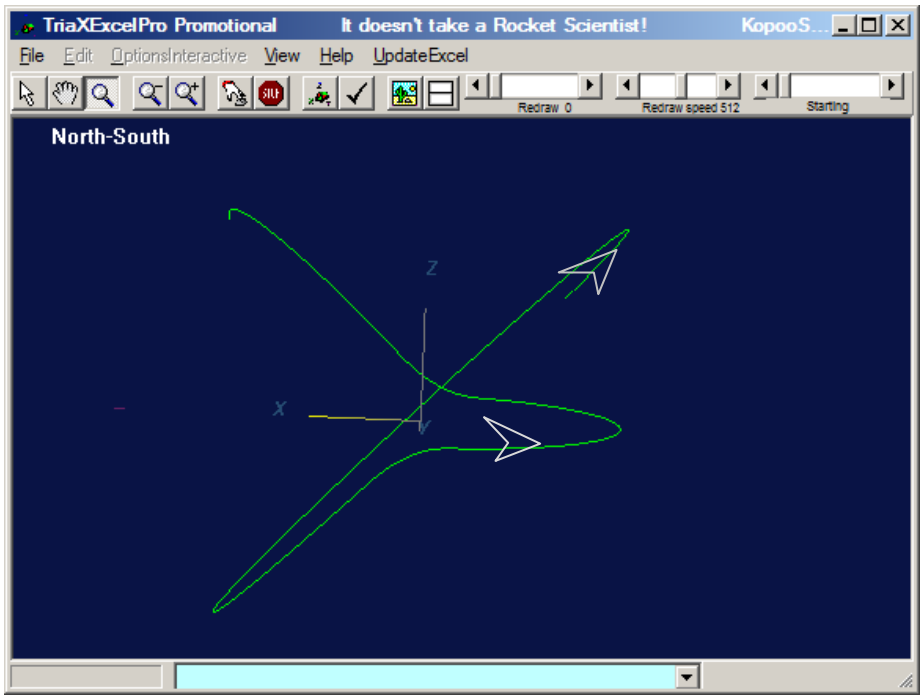


Figure 13-6: North-South station keeping with Satellite Library and 3D tool view

14. EP LIBRARY (ELECTRIC PROPULSION)

14.1 OVERVIEW

EP is an ECOSIMPRO library for the *transient* simulation of electric propulsion systems. Because the numerous possible systems, this library is more oriented as an example design for building such system. The most important features are the following:

- Thrusters' model provided with tabulated example of characteristic adapted to Hall Effect thrusters (PPS1350), Gridded Ion Engines (T6, EsaXX) and for UserDefined models.
- Xenon Flow Controller (XFC) based on the use of a thermothrottle for the control of the mass flow rate and Filter Units (FU) components are more oriented for Hall Effect thrusters (PPS1350), but can be used without difficulties with Gridded Ion Engines (T6, EsaXX).
- The Power Processing Unit (PPU) component allows receiving simple Tele-command list in the experiment to successively perform the operations.

The general design of the components of EP LIBRARY is coming from open bibliography references. The user is encouraged to modify the design in order to match with its own performances and characteristics.

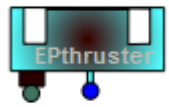
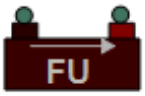
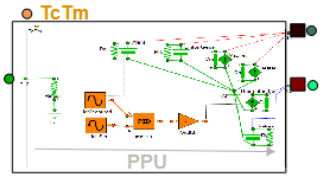
The EP components can be connected to FLUID_FLOW_1D components and to ELECTRICAL components with the aim of simulate a complete Electric propulsion system.

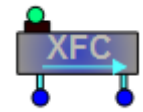
The references used for this library and components are RD-60 to RD-75

14.2 OPERATIONAL COMPONENTS

All the components of the EP Library are shown in the following table.

Table 14-1 – Components of EP Library

SYMBOL	COMPONENTS	DESCRIPTION
	EPtruster	is based on a Time dependent boundary condition in P-T; simulate the thruster performances
	FU	simulate a Filter unit for HET thrusters
	Fail_Processor	failure provided by an Enum Name in rams.event to be Set or reset by rams.set
	Connect	Simple connection with junction and volume fluid ports, but on request simulates leaks that reduce the pressure of the Volume port, else nothing
	PPU	based internally on several CONTROL and ELECTRICAL components; simulate the hardware and logic of a PPU

SYMBOL	COMPONENTS	DESCRIPTION
	XFC	is based on a AbstractJunction; provide the propellant flow according to a addimensional tables

14.2.1 Enumerative Description

In order to facilitate the reading of the equations and events, the components of EP Library relies on 5 ENUM according to the following table.

Table 14-2 – Enumerations list of EP Library

ENUM **PPU_MODES**	{Power_Off,StandBy,Configuration,Remote,Automatic,Venting}	Modes in the PPU Petri
ENUM **TC_LIST**	{None,TCStandBy,TCConfiguration,TCRemote,TCAutomatic,TCVenting,TCrequestTm,TCIdSet,TCImagnet,TCIheater,TCIthermthrottle,TCVaccel,TCValvesOn,TCValvesOff,TCLoopOn,TCLoopOff,TCIgnitorOn}	List of TC
ENUM **TM_LIST**	{TMthermistance,TMId,TMVd,TMItt,TMMode}	List of TM
ENUM **TYPE_FAILURE**	{Fail_PPU_power150pc,Fail_PPU_PROM_checksum,Fail_Cathode_operation_delay_not_achieved,Fail_Anode_current_TM_not_available,Fail_Max_Swirl_Torque,Fail_Wrong_Swirl_Orientation,Fail_XeLeakage_1,Fail_XeLeakage_2,NoFail}	List of Failures
ENUM **ThrusterTYPE**	{UserDefined,AlternateModel,PPS1350,T6,EsaXX}	List of possible thrusters

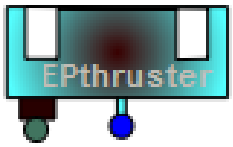
14.2.2 EPthruster

14.2.2.1 Description

The thruster operates using xenon as a propellant. Xenon atoms are ionized, then accelerated by an electromagnetic field, neutralized by electrons to maintain charge balance and finally ejected, providing the thrust, see annex 6. The fundamental parameters of any EP thruster are: thrust, mass flow rate, power.

This component simulates an electric thruster. As a fluid component it is based on boundary device of FLUID_FLOW_1D.

It is provided with a set of tables for the performance characteristics of thrusters described in the bibliography. Contrary to the Chemical propulsion, the mass flow rate of propellant is not controlled by the thrusters (there are no sonic throat equation in the operational domain of the thruster). But the current is in close relation with the mass flow rate.



14.2.2.2 Construction Parameters

None

14.2.2.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
PPUFU	ThrusterHV		IN	to connect the PPU High Voltage devices for the Thruster
f	FLUID_FLOW_1D.fluid		IN	Inlet/Outlet fluid ports

Where the PORT ThrusterHV is defined by:

NAME	TYPE	DESCRIPTION	UNITS
Idischarge	SUM REAL	Discharge Current	A
Iheater	SUM REAL	Heater Current	A
Imagnet	SUM REAL	Magnet Current	A
Ioscillation	SUM REAL	Oscillation Current	A
Power	SUM REAL	Total electrical power used at this port	W
Vacceleration	EQUAL REAL	Acceleration grid	V
Vdischarge	EQUAL REAL	Discharge or Beam Voltage	V
VignitorKeeper	EQUAL REAL	Ignitor or Keeper Voltage	V

14.2.2.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
DataThr[4,3]	REAL	{ { 5418, 15510, 14610} , { ...	PPS1350 T6 and EsaXX: goIspLoss,etaF,kI,eta_e	-
NeededHeatingTime	REAL	100	Needed heating time (s), example value	s
Rheater	REAL	0.5	Heater resistance	ohm
Rmagnet	REAL	0.5	Magnet resistance	ohm
ThrType	ENUM ThrusterTYPE	UserDefined	Definition of the electric thruster used	-
Tswirl_max	REAL	0.00014	Maximum value of swirl torque (N*m)	Nm
Tswirl_min	REAL	3e-005	Minimun value of swirl torque (N*m)	Nm
Tswirl_nom	REAL	7e-005	Nominal value of swirl torque (N*m)	Nm
VminOp	REAL	50	Minimum anode voltage for producing positive thrust, example low value	V
etaF_o	REAL	0.5815	Force efficiency for UserDefined models only	-
eta_e_o	REAL	0.8468	Electrical efficiency for UserDefined models only	-
f0	REAL	0	coefficients for the laws of F for AlternateModel models only	N
f1	REAL	1 / 14142.1	idem	N/W
f2	REAL	0	idem	N/W^2
goIspLoss_o	REAL	5418	Losses for UserDefined models only	m/s
kI_o	REAL	1.5e-006	Proportional law mfr=kI*Id for UserDefined and AlternateModel models	-
m_min	REAL	1e-007	Minimum mass flow for producing	kg/s

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
			ignition, example low value	
phi_TAP	REAL	0	Spherical coordinate phi of the thrust application point	deg
phi_dir	REAL	-90	Spherical coordinate phi or azimuth angle of thrust vector direction	deg
r_TAP	REAL	0	Spherical coordinate r of the thrust application point	m
theta_TAP	REAL	0	Spherical coordinate theta of the thrust application point	deg
theta_dir	REAL	90	Spherical coordinate theta or elevation angle of thrust vector direction	deg

14.2.2.5 DECLS (with non-default values for inherited component):

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
AlreadyHeated	BOOLEAN	FALSE	Flag true when the heating of the hollow cathode has been completed	-
AlreadyIgnited	BOOLEAN	FALSE	Flag true when ignition of the thruster has been done successfully	-
F	REAL		Thrust	N
Fx	REAL		Component X of thrust vector	N
Fy	REAL		Component Y of thrust vector	N
Fz	REAL		Component Z of thrust vector	N
Id	REAL		Actual discharge current also ion Beam current	A
IgnitionPulseCount	INTEGER	0	Current number of ignition pulses	-
Isp	REAL		Specific impulse	s
LocalTime	REAL	0	for discrete events	s
MXe	CONST REAL	0.13129 / 6.022e+023	Mass of one Xe atom (kg)	kg
NeededIgnitionPulses	INTEGER	1	Number of ignition pulses needed for ignition	
P	REAL		Pressure (Pa)	Pa
P_nc	REAL		Non-condensable partial pressure (Pa)	Pa
P_o	REAL	100000	Initial pressure (Pa)	Pa
Power	REAL		Total input thruster power i.e. Discharge power	W
T	REAL		Temperature of the fluid (K)	K
T_o	REAL	293.15	Initial temperature (K)	K
T_swirl	REAL		Swirl Torque	Nm
Th_start	REAL	1e+040	Time for beginning of heating	s
Tx	REAL		Component X of thrust torque	Nm
Ty	REAL		Component Y of thrust torque	Nm
Tz	REAL		Component Z of thrust torque	Nm
Vacceleration	REAL		Acceleration grid voltage	V
Vd	REAL		Discharge voltage also Beam voltage	V
alpha	REAL		Vapour void fraction (-)	-
cond	REAL		Thermal conductivity of fluid volume (W/m*K)	W/m*K
etaF	REAL		idem	-
eta_e	REAL		idem	-
go	CONST REAL	9.80665	the go value for the propulsion = Standard gravity of General Conference on Weights and Measures	m/s^2

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
goIspLoss	REAL		current values selected	m/s
iCode	INTEGER	0	accessory variable	-
ier	INTEGER			
ier1	INTEGER			
ipx	INTEGER		last index for x interpolation	
ipy	INTEGER		last index for y interpolation	
kI	REAL		idem	-
m	REAL		Xenon mass flow	kg/s
pi	CONST REAL	3.14159 2653589 79	pi, Archimedes' number	-
q	CONST REAL	1.609e-019	Electrical charge of one electron (C)	C
rad	REAL	pi / 180		-
rho	REAL		Density of the tank (kg/m^3)	kg/m^3
rho_o	REAL	1	Initial density (kg/m^3)	kg/m^3
rx	REAL		X position of thrust aplication point	m
ry	REAL		Y position of thrust aplication point	m
rz	REAL		Z position of thrust aplication point	m
u	REAL		Specific internal energy (J/kg)	J/kg
vel	REAL		Average fluid velocity (m/s)	m/s
visc	REAL		Viscosity of the fluid volume (Pa*s)	Pa*s
vsound	REAL		Speed of sound (m/s)	m/s
x	REAL		Vapour mass fraction (-)	-
x_o	REAL	0	Initial quality (-)	-

14.2.2.6 FORMULATION:

Every model is based from correlations formulae, but those formulae are customizable: Users who want to use other thrusters have to adjust the parameters available in order to fit with the integrated formulations.

General model

The model used for the relation Thrust Power and Isp is developed in annex.

This is a simplified model based on 2 parameters (η_F and $g_o Isp_{loss}$), but valid for Hall and Ion thrusters.

1. $F = \dfrac{\eta_F \cdot P_e}{\frac{1}{2}(g_0.Isp)^2 + (g_0 Isp_{Loss})^2} \cdot g_0.Isp$ with F in N, P_e in W and $g_0.Isp$ in m/s
2. With a given total mass flow rate (kg/s) and F the $g_0.Isp$ (m/s) can be deduced $g_0.Isp = F / \dot{m}_{prop}$ thus the two equations 1) and 2) make the variable $g_0.Isp$ as algebraic and thus the two equations 1) and 2) solved in a box

In addition the current (discharge current or beam current) is derived from the total mass flow rate by a relation with one coefficient k_I (theoretically one electron for one ion) which is actually also linked to the fraction of propellant feeding the anode with respect to the total mass flow rate:

3. $I_d = k_I / \dot{m}_{prop}$ with I_d in A

And the voltage (discharge voltage or beam voltage) is derived from the total input power and current by a relation with one coefficient η_e:

4. $U_d = \eta_e \cdot P_e / I_d$ with U_d in V

The following table presents the proposed coefficients for 3 thrusters:

Table 14-3 – General model: proposed coefficients for 3 thrusters

Coefficents\Thruster	PPS1350	T6	EsaXX	Comments
goIspLoss (m/s)	5418	15510	14610	Estimation (from public data)
η_F	58.15%	99.25%	89.0%	Estimation (from public data)
k_I (kg.s^{-1}.A^{-1})	1,608E-06	1,56E-06	1,534E-06	Example
η_e	100%	89%	85%	Example for T6 and EsaXX,

Two last equations are coming from the fluid port that impose the mass flow rate and from the PPU that impose the voltage

5. $\dot{m}_{prop} = given\_by\_the\_XFC$
6. $U_d = given\_by\_the\_PPU$

Alternative model

For some application a model based on polynomial degree of 2 of the total input thruster power can be used:

1. $F = f_2 P_e^2 + f_1 P_e + f_0$ with F in N and P_e in W
2. $\dot{m}_{prop} = m_2 P_e^2 + m_1 P_e + m_0$ with $\dot{m}_{prop}$ in kg/s
3. $\dot{m}_{prop} = k_I \cdot I_d$ with I_d in A
   (and where k_I can be set from an equation like $\dot{m}_{prop} = m_2 P_e^2 + m_1 P_e + m_0$ if available)
4. $U_d = \eta_e \cdot P_e / I_d$ with U_d in V

Note: For the two models, the value of Id provided by those equations are a consequence of the two coefficients kI and η_e : for more realistic results, those should be trimmed around the wanted operational point.

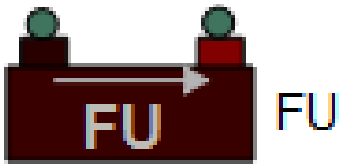
As before, two last equations are coming from the fluid port that impose the mass flow rate and from the PPU that impose the voltage

5. $\dot{m}_{prop} = given\_by\_the\_XFC$
6. $U_d = given\_by\_the\_PPU$

14.2.3 FilterUnit

14.2.3.1 Description

The Filter Unit (FU) is inserted between the Hall Effect thrusters and the PPUs to protect the latter from electrostatic discharges or EMI and to reduce plasma discharge current oscillations. The fundamental parameters of a FU are: the RMS value of the discharge current feeding the thruster (FUosc).



14.2.3.2 Construction Parameters

None

14.2.3.3 PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
PPU	ThrusterHV		IN	to connect the PPU High Voltage devices for the Thruster
THR	ThrusterHV		OUT	to connect the Thruster High Voltage

14.2.3.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
CoeffIdHighVoltage	REAL	0.2	example FUosc= CoeffIdHighVoltage*Id	-
CoeffIdLowVoltage	REAL	0.25	example FUosc= CoeffIdLowVoltage*Id	-
CurrentTransition	REAL	2	example Ioscillation= 0 before	A
Requiv	REAL	0.1	example of internal resistance for dissipation	ohm
VoltageTransition	REAL	250	example	V

14.2.3.5 FORMULATION:

This component is simulated by a simple set of bidirectional connections from PPU to THR. In addition as done for some thrusters of HET type, the FU output the oscillation current. The example law is based on two cases for low current and low voltage with an output set to "Id. CoeffIdLowVoltage" and else an output set to "Id. CoeffIdHighVoltage"

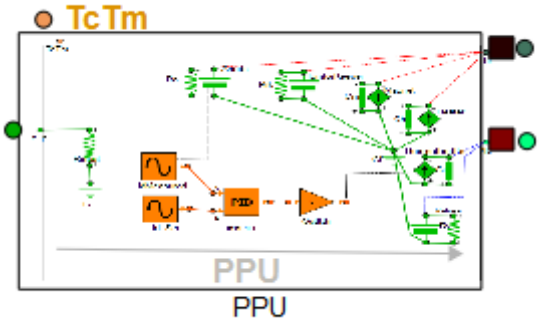
The thermal dissipation power is based on a simplified joule effect with a résistance Requiv

14.2.4 PPU

14.2.4.1 Description

The description of the PPU has two parts: Hardware and Software.

- The hardware part contains all the devices that are used in the PPU based internally on several CONTROL and ELECTRICAL components
- The software part is dealing with logic of operations managed into the PPU.



14.2.4.2 Construction Parameters

None

14.2.4.3 PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
TcTm	TCTM		IN	Communication direct and digital
Thr	ThrusterHV		OUT	to connect the Thruster High Voltage devices or FU
Xfc	ThrusterLV		OUT	to connect the Thruster Low Voltage devices: XFC
e_p	PORTS_LIB.elec		IN	Connection to the main power bus

14.2.4.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
ImaxOp	REAL	5	Operational domain at constant voltage before the knee	A
IminOp	REAL	1	Operational domain with loop control	A
Ishort	REAL	9	Short circuit Current after the knee	A
Ittmax	REAL	4	Thermothrottle maximum current	A
PulseWidth	REAL	0.005	Duration of ignition pulses	s
StandByPower	REAL	20	power consumption of the PPU in stand-by mode	W
Tflow	REAL	5	Duration of flow while still heating cathode before ignition	s
Theating	REAL	120	Duration of heating cathode before ignition	s
ThrType	ENUM ThrusterTYPE	UserDefined	See § Enumerative Description	-
Tventing	REAL	10000	Duration of venting operation	s
VignitorKeeper	REAL	280	Ignition voltage	V
VminOp	REAL	100	Operational domain	V
Vvalves	REAL	28	Valves voltage	V
e0	REAL	0.05	Coefficient losses for the Electrical efficiency of the PPU: etaPowerPPU=1-(e0+e1*PowerOutPPU+e2*PowerOutPPU**2)	-

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
e1	REAL	1e-005	Coefficient losses for the Electrical efficiency of the PPU	-/W
e2	REAL	0	Coefficient losses for the Electrical efficiency of the PPU	-/W^2

14.2.4.5 DECLS:

NAME	TYPE	DESCRIPTION	UNITS
AnodeSupply	REAL	1: Enable the supply 0: disabled. Those are activated by events, so discrete variables	
BusPowerON	BOOLEAN		
HeaterSupply	REAL		
Ibus	REAL	Current used by PPU on the bus	
Id	REAL	Discharge current	A
IdSet	REAL	Default Discharge Current Set to be controlled	A
IgnitionPulse	BOOLEAN		
IgnitorSupply	REAL		
Ih	REAL	Heater current	A
IhSet	REAL	Default Heater current sourcing level	A
Im	REAL	Magnet current	A
ImSet	REAL	Default Magnet Current Set	A
Iosc	REAL	Thruster RMS oscillation current measured into the FU	A
Itt	REAL	Thermothrottle current	A
IttOpenLoop	REAL	Thermothrottle current in open loop	A
IttSet	REAL	Default Thermothrottle Current Set	A
LoopON	BOOLEAN	Enable the loop control	
MagnetSupply	REAL		
Mode	ENUM PPU_MODES	See § Enumerative Description	
PPU_ON	BOOLEAN		
PowerInPPU	REAL	Power used by the PPU on the bus	W
PowerOutPPU	REAL	Power delivered by the PPU to the EP devices (thruster, XFC, FU)	W
SlopeAfterKnee	REAL	negative or null resistance by design	ohm
TCforbidden	REAL	Flag for the User	-
ThermothrottleSupply	REAL		
Vaccel	REAL	Acceleration grid voltage	V
VaccelSet	REAL	Default Accel voltage Set	V
ValveSupply	REAL		
Vd	REAL	Anode discharge voltage	V
VdSet	REAL	Default Anode voltage Set	V
VectorIttmax[1]	REAL		
Vik	REAL	ignition voltage	V
Vv	REAL	valves voltage	V
etaPowerPPU	REAL	Electrical efficiency of the PPU	

14.2.4.6 COMPONENT INSTANCES (TOPOLOGY BLOCK)

The PPU comprises components from other ESPSS libraries as sketches below:

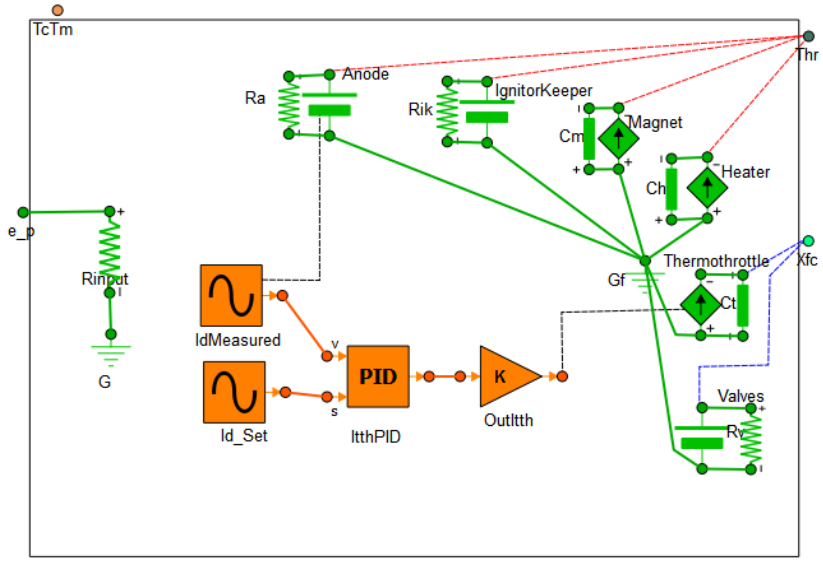


Figure 14-1: PPU topology from ELECTRICAL and CONTROL libraries

OBJECT	COMPONENT TYPE	DESCRIPTION
IdMeasured	CONTROL.AnalogSource	
Id_Set	CONTROL.AnalogSource	
ItthPID	CONTROL.Cntrl_PID	
OutItth	CONTROL.Gain	
Rinput	ELECTRICAL.Resistor	
Gf	ELECTRICAL.Ground	
Anode	ELECTRICAL.VoltageConstant	
Valves	ELECTRICAL.VoltageConstant	
Thermothrottle	ELECTRICAL.CurrentConstant	
G	ELECTRICAL.Ground	
Heater	ELECTRICAL.CurrentConstant	
Magnet	ELECTRICAL.CurrentConstant	
IgnitorKeeper	ELECTRICAL.VoltageConstant	
Ra	ELECTRICAL.Resistor	
Rv	ELECTRICAL.Resistor	
Rik	ELECTRICAL.Resistor	
Ch	ELECTRICAL.Conductor	
Ct	ELECTRICAL.Conductor	
Cm	ELECTRICAL.Conductor	

14.2.4.7 COMPONENT INSTANCES DATA (with non-default values):

DATUM	VALUE	TYPE	DESCRIPTION	UNITS
Anode.V	Vd	REAL	Value of constant voltage	"V"
Heater.I	Ih	REAL	Value of constant Current	"A"
IdMeasured.Amp	Id	REAL	Signal amplitude or height	"-"
Id_Set.Amp	IdSet	REAL	Signal amplitude or height	"-"
IgnitorKeeper.V	Vik	REAL	Value of constant voltage	"V"
ItthPID.Ti[n]	10	REAL	Integrator time or reset time	"s"

ItthPID.end_pos	end_PI	ENUM EndPosBehaviour	End position behaviour	"-"
ItthPID.u_max[n]	VectorIttmax	REAL	High limit of output	"-"
Magnet.I	Im	REAL	Value of constant Current	"A"
Ra.R	1000000	REAL	Resistance	"Ohm"
Rik.R	1000000	REAL	Resistance	"Ohm"
Rinput.R	1.66	REAL	Resistance	"Ohm"
Rv.R	1000000	REAL	Resistance	"Ohm"
Thermothrottle.I	Itt	REAL	Value of constant Current	"A"
Valves.V	Vv	REAL	Value of constant voltage	"V"

14.2.4.8 FORMULATION:

14.2.4.8.1 Hardware

From the block diagrams found in the bibliography, the PPU (or PSCU) component shall include 5 power supplies.

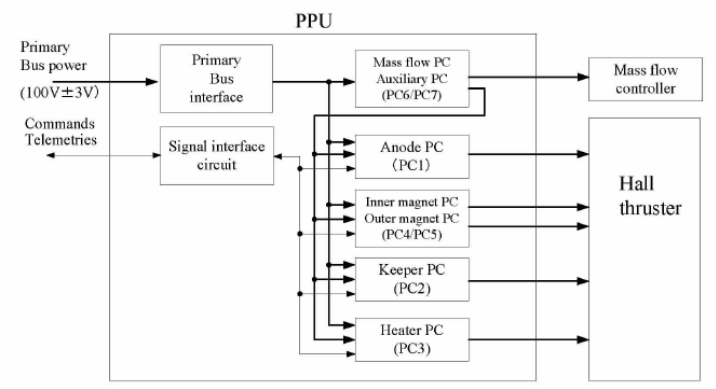


Figure 14-2: Example PPU block diagram

All the current sources or voltage sources are simulated by standard components except the main power source (Anode power source) which includes a particular important feature (needed for some Hall Effect thrusters) with a limited power discharge voltage-current characteristic.

Moreover, the control loop of the discharge current is active only in the domain grayed below in the following figure.

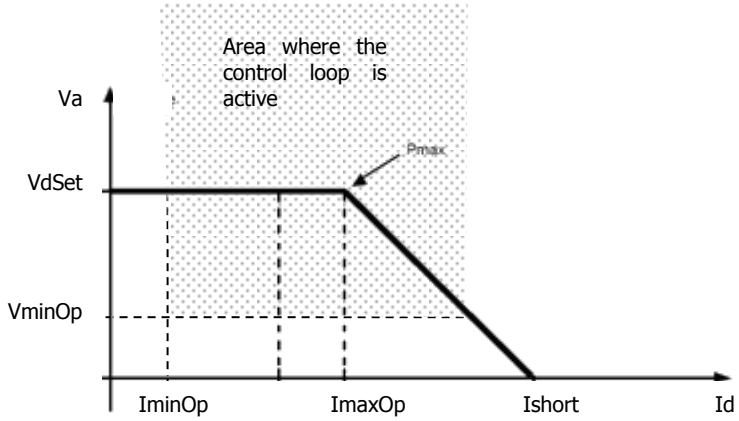


Figure 14-3: Example PPU voltage-current characteristic

The simple equation to model this power source used is

$$V_d = V_{dSet} + a(I_d - I_{\max Op})$$

where *a*=negative infinite for GIE; and for HET a <0. For V_d=0, the current is defined by I_{short} that is set in the data; the coefficient *a* is named "SlopeAfterKnee": that is a negative resistance (in ohm)

- The other power supplies for feeding the devices included in the thruster are:
  Magnet power supply,
  Igniter/ Keeper power supply,
  Heater power supply,
  Acceleration grid power supply.
- A last power supply with control loop for the mass flow rate
  In the case of Hall Effect thrusters (or thruster using thermal flow controller), this is a current source where the current level is given by the error between the actual discharge current and the goal I_{dSet} given in the data of the component.
  For initialization purpose, when the thruster is still off (Id=0) the current level of the current source shall be set to an initial current IttSet (sometime called warm-up current $I_{ttWarmUp}$).

  The control loop may use according to the gains Kp, Ki, Kd a proportional loop control, or a PI loop control or full PID loop control:

  The PID implemented in the PPU is a standard PID coming from the CONTROL Library with most of the default settings. The current position corresponds to the Id measured while the desired position is IdSet. The output toward the thermothrottle is the current Itt.

  - Because the increase of Itt decreases the mass flow rate and thus decreases the current Id, the gain of the controller shall be negative (-1).
  - The time constant of integration has been set to 10 s in order to follow approximately some real devices.
  - The range of output for Itt has been set to the maximum Itt "Ittmax"current defined in the data.

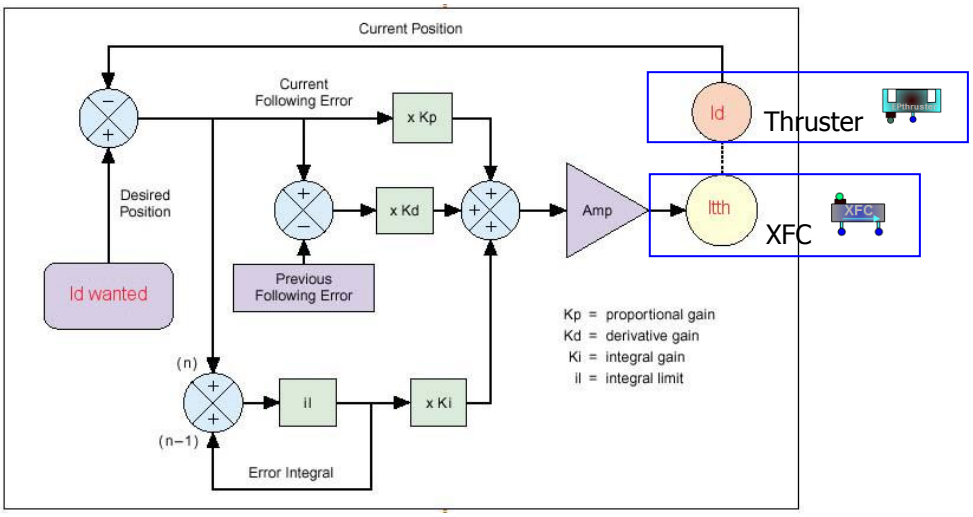


Figure 14-4: Control loop of the thruster discharge current

In the case of Ion engine, if different from above, the user shall use the available components of the ESPSS library (heaters, proportional valves) to build the relevant control loop because up to now, the details of the control for those thrusters are not disclosed. But it must be highlighted that the HET XFC works also with Ion engines.

14.2.4.8.2 Software

For Hall Effect thrusters the following accurate description available in the bibliography is taken into account:

The PPUs exchange data with the on-board computer. According to the received command, the PPU enters into one of its six possible modes: OFF, STAND-BY, CONFIGURATION, VENTING, AUTOMATIC and REMOTE. These transitions are as follows:

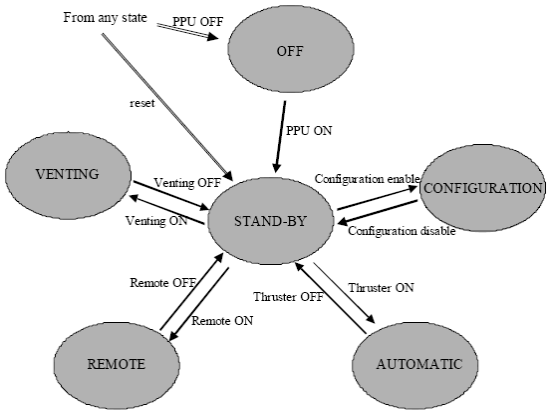


Figure 14-5: PPU Modes for HET

1. After the power bus of the satellite is connected to the PPU (switch HPSSC closed for example) the PPU is still OFF but it can receive only direct telecommand and act accordingly.
2. Then the user send the DTC (direct telecommand) "PPU ON" to turn ON the PPU DC/DC, the sequencer enters into an initialisation phase before being put in STAND-BY mode.
   During this initialisation, the ckecksum of the ROM memory is performed. If a failure is injected for this check, the PPU does not start.
3. In the "STAND-BY" mode, only low level electronics are active. But it is also possible to modify the setting of firing parameters (mode independent parameters).
   From the "STAND-BY" mode, the PPU accepts any transition to another mode and the entire mode independent TC.
   If a wrong TC is commanded from this mode, the PPU output a flag "TCforbidden".
4. In the "CONFIGURATION" mode the PPU accepts command for example to turn on the XFC valves. The user shall send "TCStandBy" to exit that mode.
5. The "VENTING" mode is only used at the mission beginning to allow venting of the Xenon lines before the first SPT firing. In this mode the PPU opens all the XFCs valves on the selected thrusters for the predetermined duration defined in the data by "Tventing".
6. In the "REMOTE" mode the PPU accepts dedicated commands. The on-board computer allows "step-by-step" management of the thrusters in order to provide flexibility to the ground operators if necessary: to turn on or off individually every power supplies and to perform the ignition pulses and to turn on or off the loop control.
7. The "AUTOMATIC" mode is the nominal mode of the PPU when the PPU drives the firing of the thruster in closed-loop.
   The power supplies of the PPU are active. The firing is driven with the set of parameters defined in the configuration mode.

   The logic of the "AUTOMATIC" mode is as follows:

   ➢ During the starting phase, the PPU sequences as follows:

   - Heat the selected cathode and thermothrottle up to the corresponding stabilised thrust level for the thruster

   - Open the XFC valves

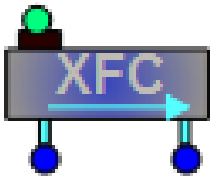
   - Supply the cathode with ignition pulses once the cathode is enough heated (according to a timer > Theating)

- The HET has to be started within a given delay period from the start of this sequence. In case of failure the PPU act accordingly.
- During the stabilised phase (after the thruster starting sequence), the initiator electrode is no longer supplied and the closed control loop becomes operational. In stabilised mode the discharge current control is made by adjusting the Xenon flow rate through the thermothrottle in the XFC

14.2.5 XFC

14.2.5.1 Description

This component (also called Flow Control Unit FCU) can be explicitly developed with a real thermothrottle component (to be derived from a tube and its envelope), but such design requires too many input data, moreover such date are generally only available in house of the thruster manufacturer. Hence, as for the thruster model, the XFC model has been derived from correlations. A general XFC characteristic is given from the bibliography. This component is based on a AbstractJunction and provides the propellant flow according to a addimensional tables



14.2.5.2 Construction Parameters

None

14.2.5.3 PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
f1	FLUID_FLOW_1D.fluid		OUT	Inlet/Outlet fluid port number 1
f2	FLUID_FLOW_1D.fluid		OUT	Inlet/Outlet fluid port number 2
xfc	ThrusterLV		IN	to connect the PPU Low Voltage devices for XFC

14.2.5.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
NominalItt	REAL	2.4	Nominal Thermothrottle current; full range 0 to 2.5*NominalItt; used for Ittr = Itt/NominalItt	A
NominalMfr	REAL	5e-006	Nominal mass flow rate; in cold case full range (0.5 to 2)*NominalMfr for mfrr = mfr/NominalMfr	kg/s
NominalPXe_bar	REAL	2.65	Nominal Xenon pressure	bar
Rtth	REAL	0.5	example of internal resistance for dissipation	ohm
Rvalves	REAL	200	example of internal resistance for dissipation	ohm
Vvalves_drop_out	REAL	10	Valve nominal drop out voltage	V
XFC_Tk	REAL	300	Temperature of XFC body	K
n	REAL	2	Exponent of pressure for the mfr law	-
tau_tt	REAL	2	Time constant for the thermothrottle	s
x_jun	REAL	0	Junction X coordinate relative to a body axis system (m)	m
y_jun	REAL	0	Junction Y coordinate relative to a body axis system (m)	m

z_jun	REAL	0	Junction elevation relative to a body axis system (m)	m

NAME	TYPE	DESCRIPTION	UNITS
NominalItt	REAL	Nominal Thermothrottle current; full range 0 to 2.5*NominalItt; used for Ittr = Itt/NominalItt	A
NominalMfr	REAL	Nominal mass flow rate; in cold case full range (0.5 to 2)*NominalMfr for mfrr = mfr/NominalMfr	kg/s
NominalPXe_bar	REAL	Nominal Xenon pressure	bar
Rtth	REAL	example of internal resistance for dissipation	ohm
Rvalves	REAL	example of internal resistance for dissipation	ohm
Vvalves_drop_out	REAL	Valve nominal drop out voltage	V
XFC_Tk	REAL	Temperature of XFC body	K
n	REAL	Exponent of pressure for the mfr law	-
tau_tt	REAL	Time constant for the thermothrottle	s
x_jun	REAL	Junction X coordinate relative to a body axis system (m)	m
y_jun	REAL	Junction Y coordinate relative to a body axis system (m)	m
z_jun	REAL	Junction elevation relative to a body axis system (m)	m

14.2.5.5 DECLS (with non-default values for inherited component)::

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
Itt	REAL		Thermothrottle current	A
PXe	REAL		Xenon pressure	bar
ValvesPosition	REAL	0	Position of the poppet with respect to the seat (equivalent to a stroke) 1 for open; 0 valve is closed	-
m	REAL		Mass flow - positive from f1 to f2 (kg/s)	kg/s
mfr_r	REAL		Addimensional Mass flow; mfr ratio = mfr/NominalMfr	-
mfrr_vs_temp_Ittr	TABLE_2D		Table of addimensional performance	

14.2.5.6 FORMULATION:

For allowing a full authority around the nominal mass flow rate at hot conditions, the mass flow rate ratio is set to 1 for the middle range of mass flow for the most constraining conditions i.e. for the hot case. But the thermothrottle current ratio is set to 1 for the most usual conditions i.e. for the ambient case. With such rule, the normal operation of the XFC control can start at ambient condition with the nominal mass flow rate, and in the event the temperature of the device increase, then the corresponding thermothrottle current shall decrease for maintaining the mass flow rate.

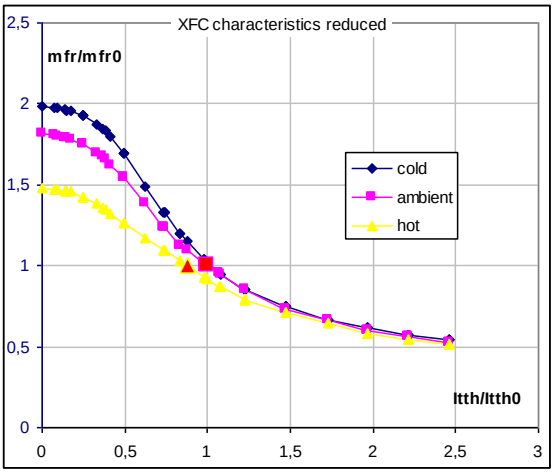


Figure 14-6: XFC thermothrottle addimentional characteristics from bibliography

Such curves of mass flow rate ratio versus the thermothrottle current ratio are put in a Table 2D function of the temperature condition case and the thermothrottle current ratio:

TABLE_2D mfrr_vs_temp_Ittr { { 263 ,293 , 333 } , {0 , 0.08 , 0.09 , 0.13 , 0.15 , 0.17 , 0.24 , 0.25 , 0.33 , 0.36 , 0.38 , 0.41 , 0.49 , 0.62 , 0.74 , 0.74 , 0.83 , 0.87 , 0.98 , 0.99 , 1 , 1.07 , 1.08 , 1.22 , 1.23 , 1.47 , 1.73 , 1.97 , 2.21 , 2.46 } ,

{ {1.984 , 1.977 , 1.976 , 1.965 , 1.96 , 1.953 , 1.929 , 1.928 , 1.869 , 1.845 , 1.833 , 1.793 , 1.699 , 1.49 , 1.332 , 1.331 , 1.195 , 1.151 , 1.036 , 1.031 , 1.019 , 0.956 , 0.949 , 0.856 , 0.853 , 0.749 , 0.661 , 0.614 , 0.574 , 0.542 } ,{1.817 , 1.803 , 1.801 , 1.793 , 1.787 , 1.779 , 1.753 , 1.753 , 1.697 , 1.673 , 1.656 , 1.624 , 1.547 , 1.383 , 1.236 , 1.235 , 1.124 , 1.092 , 1.012 , 1.008 , 1 , 0.956 , 0.949 , 0.853 , 0.85 , 0.733 , 0.661 , 0.598 , 0.558 , 0.526 } ,{1.482 , 1.474 , 1.472 , 1.465 , 1.462 , 1.458 , 1.426 , 1.426 , 1.386 , 1.36 , 1.347 , 1.323 , 1.267 , 1.175 , 1.092 , 1.091 , 1.028 , 1 , 0.928 , 0.924 , 0.917 , 0.875 , 0.869 , 0.792 , 0.789 , 0.709 , 0.645 , 0.582 , 0.542 , 0.518 } } }

The ambient condition characteristic is be used. If the current temperature of the XFC is specified (in Kelvin), linear interpolation with the temperature case (263; 293; 333 K).

The effect of the pressure is taken into account by a pressure ratio along with an exponent (set by default to n=2).

Finally the XFC model is described by the following equation:

$$\dot{m} = (P/P_{nominal})^{n}.mfrr(T\_K, Ittr).mfr_{nominal}$$

with $Itt/Itt_{nominal}$ = Ittr and where $P_{nominal}$, $Itt_{nominal}$ and $mfr_{nominal}$ are set in the data of the component.

14.2.6 Fail_Processor

14.2.6.1 Description

This component receives failure cases provided by the user. It manages the setting or the clearing of a failure, and save the current failure case in a global variable.

14.2.6.2 Construction Parameters

None

14.2.6.3 PORTS

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
rams	RAMS		IN	

Where the PORT RAMS is defined by:

NAME	TYPE	DESCRIPTION	UNITS
event	EQUAL ENUM TYPE_FAILURE	Name of a failure case	-
set	EQUAL BOOLEAN	TRUE to Set a failure; FALSE to undo (reset) the failure	-

And the ENUM TYPE_FAILURE is defined by:

{Fail_PPU_power150pc,Fail_PPU_PROM_checksum,Fail_Anode_current_TM_not_available ,Fail_Max_Swirl_Torque,Fail_Wrong_Swirl_Orientation,Fail_XeLeakage_1,Fail_XeLeakage _2,NoFail}

14.2.6.4 FORMULATION:

The component act first on the detection of a new failure case on its port "rams.event" where the failure case are in clear name thanks to the enumeration list. But the action to reset (clear) a failure case is taken in second from the value of the Boolean port "rams.set=FALSE". Hence to set a failure case, the user shall send:

- rams. event = Fail_Anode_current_TM_not_available for example.

For resetting any failure, the user shall send:

- rams.set=FALSE

To be independent on the global variable (keeping in memory the value of the current failure case), a specific Boolean function has been added named "event_RAMS(x)" where in single parameter "x" the failure case is in clear name thanks to the enumeration list.

Hence the components specific actions according to the failure case considered can be written with ZONE statements like

T_swirl = ZONE (event_RAMS (Fail_Max_Swirl_Torque)) Tswirl_max …

14.2.7 Connect

14.2.7.1 Description

Simple connection with junction and volume fluid ports, but on request simulates leaks that reduce the pressure of the Volume port. Thanks to the use of the two ports of type junction and volume the component can be inserted in any fluid schematic.

14.2.7.2 Construction Parameters

None

14.2.7.3 PORTS:

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
f1	FLUID_FLOW_1D.fluid		OUT	Junction port
f2	FLUID_FLOW_1D.fluid		IN	Volume port

14.2.7.4 DATA:

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
XeLeakageRatio_1	REAL	0.9	ratio of pressure of the Volume port/ P Junction port	-

XeLeakageRatio_2	REAL	0.1	ratio of pressure of the Volume port/ P Junction port	-

14.2.7.5 FORMULATION:

According to the values of the leakages ratio set in the data, the pressure of the volume port is reduced with respect to the junction port.

Important note:

The mass flow rate in the component is not modified, thus the leakage is only simulated by a reduction of the pressure in the volume port.

The reason of this choice come from simulation duration that is much faster than when using as sketched at right a real leakage system using tubes, tee, valve and boundary condition

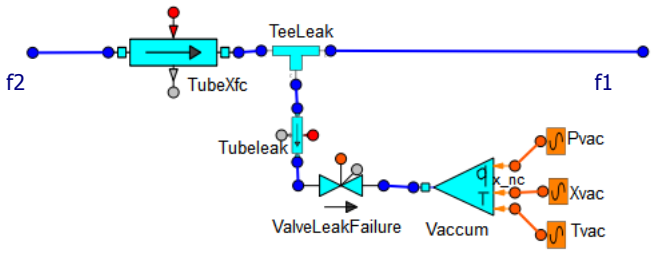


Figure 14-7: Other possible Connect component

14.3 FUNCTIONS

Two functions are added to the library

1. A General function providing the index number of an ENUM

   FUNCTION INTEGER Index(ENUM PPU_MODES Mode

   This function is used in the telemetry output "TcTm.TM_Data[TMMode] = Index(Mode)"

2. A General Function that check if the failure in parameter is activated

   FUNCTION BOOLEAN event_RAMS(ENUM TYPE_FAILURE FailureName

   This function, as mentioned above, is used in the management of the failure cases by each component concerned. It avoid the use of the global variable set for keeping in memory the current failure case.

14.4 EXAMPLE OF USE

A stand-alone test of every single component is barely meaningful.

However building a generic EP system has been performed

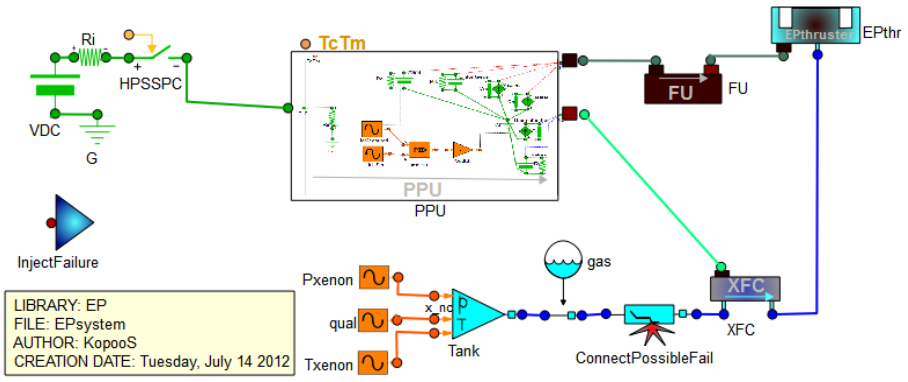


Figure 14-8: Generic EP system with default settings

The simulation run includes the operations to turn on the PPU and to send tele-commands in order to reach the automatic mode. In addition at TIME=410 s a tele-commands is sent for changing the discharge current setting to a value above the knee of anode power supply.

The following experiment produces the outputs shown in figure below.

```
        -- '23/07/2012  11:15:49

EXPERIMENT expDefault ON EPsystem.default
DECLS
  STRING RepName="ReportsMemory" --added for tracability
  OBJECTS
INIT
      -- initial values for state variables
      PPU.ItthPID.vi[1] = 0
      PPU.ItthPID.yf[1] = 0
      -- initial values for algebraics
      EPthr.F = 0
BOUNDS
BODY
    --============================================================
    -- mandatory to be set here in the experiment
      PPU.ItthPID.u_max[1] = PPU.Ittmax
    --============================================================
      --Particular settings
      VDC.V=50
      TIME = 0 -- Time shall be set before any AFTER statement
      --============================================================
      --OPERATION and TC LIST
      --============================================================
      --Turn ON the main power from the bus
      HPSSPC.b_fire.signal[1] = FALSE
      HPSSPC.b_fire.signal[1] = TRUE AFTER 0.5
      --Direct TC to turn ON the PPU
      PPU.TcTm.DTC=1 AFTER 3
      --Sequence of digital TCs in the chronological order else CANCELLED EVENTS OCCURS
      PPU.TcTm.TC=None AFTER 6 --nothing
      PPU.TcTm.TC=TCAutomatic AFTER 8 --switch to automatic Mode sequence with default settings
      PPU.TcTm.TC_Value=4.125 AFTER 12 -- prepare a TC with a value
      PPU.TcTm.TC=TCIdSet AFTER 12 -- send the TC to change the IdSet (valid in any mode)
      PPU.TcTm.TC_Value=5.125 AFTER 410 -- prepare a TC with a value
      PPU.TcTm.TC=TCIdSet AFTER 410 -- send the TC to change the IdSet (valid in any mode)
      --... etc.
      --============================================================
      RDIGITS=8
      ABS_ERROR=1E-5
      REL_ERROR=ABS_ERROR
      REPORT_MODE=IS_STEP -- REPORT_MODE=IS_EVENT,IS_CINT,IS_STEP
      -- report results in file reportAll.rpt
      RepName= "default"
      REPORT_TABLE(RepName , " *P *d *t *r TRACEABILITY *type* *fluid *T ")
      INTEG_TO(750,10) -- integrate the model
      --============================================================
END EXPERIMENT
```

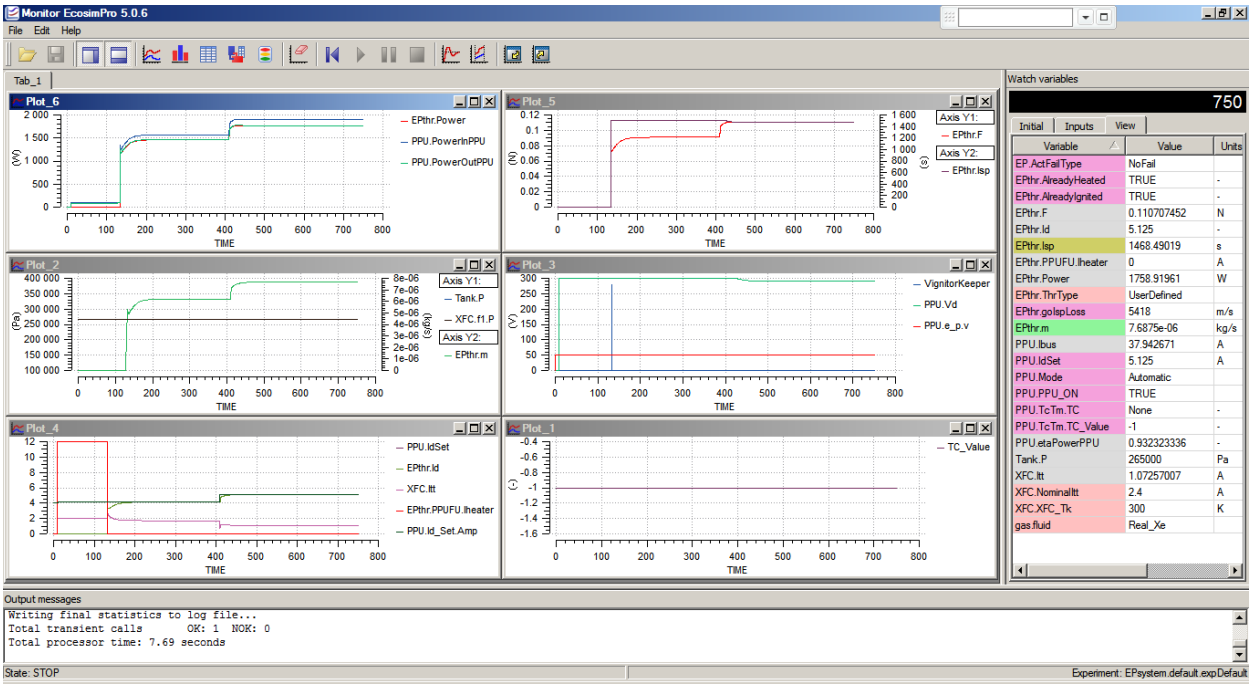


Figure 14-9: Reference simulation output for the Generic EP system with default settings

15. APPLICABLE AND REFERENCE DOCUMENTS (VOL 2)

15.1 APPLICABLE DOCUMENTS

AD-1 086-038-XY-L-X1820 "ESPSS Version 2. Management Proposal"

AD-2 086-038-XY-L-X1819 "ESPSS Version 2. Technical Proposal"

AD 3 086–031–LRU–L–0001 "ESPSS User Requirements Document"

AD 4 086–031–NT–L–0001 "ESPSS Physical Model Specification Document"

15.2 REFERENCE DOCUMENTS

RD-48 USER Manual TriaX OrbitaL tool available at www.kopoos.com

RD-49 Cours de technologie spatiale: - Techniques et technologies des véhicules spatiaux T 4, CNES, 1993.

RD-50 Michael Paluszek et al., Spacecraft attitude and orbit control, Princeton Satellite systems inc., 2009 .

RD-51 R. Ouziaux, J. Perrier, Mécanique des fluides appliquée, tome 1 Fluides incompressibles, 1966.

RD-52 EcosimPro FLIGHT_SIM Library

RD-53 P. Duchon et al., STABILISATION DES SATELLITES, SupAero, 1983.

RD-54 R. Guiziou., DESS AIR & ESPACE, systèmes de contrôle d'attitude et d'orbite, Uni. Aix-Marseille III, 2001.

RD-55 S.Bouiges, Calculs astronomiques pour amateurs, Masson, 1992.

RD-56 Vernon Chi. Quaternions and Rotations in 3-Space, 25 September 1998 ; Leandra Vicci, 27 April 2001

RD-57 Vladimir A. Chobotov, Orbital Mechanics course.

RD-58 Eai-Kci-Me-09 TN3610PhysicalModelSpec Satellite Library06 Technical note TN-3610 Physical Model Specification .

RD-59 Eai-Kci-Me-12 TN-5510 Mission Cases using EcosimPro03, Technical note TN-5510 Mission Cases

RD-60 Marek Ziebart, "Generalized Analytical Solar Radiation Pressure Modeling Algorithm for Spacecraft of Complex,shape," University College London AIAA journal Vol. 41, No. 5, September-October 2004.

RD-61 Christophe Koppel, Serge Barral, « High Specific Impulse Thrusters Behavior » Atelier Cnes "Trajectoires A Poussee Faible" 7-8 mars 2000, Toulouse.

RD-62 Library ESPSS 2.0 beta3 dated September 2009.

RD-63 Jesus Alvarez, Ramon Perez-Varra, Christophe R. Koppel "SESP 2006-435470 Simulation of the Smart-1 Electric Propulsion System With a System Simulation Software Ecosimpro®", presented at the 9th International Workshop on Simulation for European Space Programmes, 6-8 November 2006 at ESTEC, Noordwijk, the Netherlands

RD-64 Pierre Dumazert, Frédéric Marchandise, Mathieu Prioul, Franck Darnon, Laurent Jolivet AIAA 2003-4549, PPS® 1350-G , 39th Joint Propulsion Conference, Huntsville, Alabama, July 2003.

RD-65 Matthias Gollor and Simon Weinberg. AIAA 2008-5284 "Electric Propulsion Electronics Activities in Europe" European Space Agency, Michael Boss Astrium GmbH, Federico de la Cruz Astrium CRISA, Paolo Galantini Galileo Avionica S.p.A., Eric Bourguignon Thales Alenia Space ETCA, 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit 21 - 23 July 2008, Hartford, CT.

RD-66 Christophe R. Koppel, Olivier Secheresse "PERFORMANCES AND THRUST MODELISATION OF A HALL EFFECT THRUSTER" LOTUS 2 Toulouse, France, 18 – 20 juin 2002.

RD-67 C.H.McLean and J.B.McVey, " Testing of a US-Built HET System for Orbit transfer applications", 35 th AIAA/SAE/ASME/ASEE Joint Propulsion Conference and Exhibit, June 20-24, 1999 / Los Angeles.

RD-68 Brophy J.R., " Ion thruster performance model", NASA CR-174810, December 1984

RD-69 Martin Hechler, " MAS Working Paper No. 389 Mercury Orbiter Mission Analysis: On Mission Opportunities with Chemical and Solar Electric Propulsion" October 8, 1996 ESOC

RD-70 Hiroyuki Osuga et al., IEPC-2005-114 "Development Status of Power Processing Unit for 200mN-class Hall Thruster" Mitsubishi Electric Corporation

RD-71 Olivier Duchemin et al., AIAA 2010-6696 "Electric Propulsion Thruster Assembly for Small GEO"

RD-72 Jon A. Christensen et al., AIAA 99-2972 The NSTAR Ion Propulsion Subsystem for DSI Hughes Electron Dynamics, 35th AIAA / ASME / SAE / ASEE Joint Propulsion Conference & Exhibit June 20-23, 1999/Los Angeles, CA

RD-73 Howard Gray et al,. IEPC-2005-082, Inmarsat 4F1 Plasma Propulsion System Initial Flight Operations Presented at the 29th International Electric Propulsion Conference, Princeton University, October 31 – November 4, 2005

RD-74 H. Bassner, R. Bond, K. Groh, 1997ESASP.398, page 251-257 "The ESA-XX Ion Thruster", ESA Space propulsion conference, 1997.

RD-75 Moonish Patel, Neil Goodzeit, AIAA 2004-3187 "High Efficiency Flight Control System for GEO Spacecraft Hall Current Thruster Orbit Transfer," Lockheed Martin Space Systems 22nd AIAA International Communications Satellite Systems Conference & Exhibit 2004.

APPENDICES

A.1 QUATERNION'S ALGEBRA FORMULATION

A quaternion algebra can be described as a 4-dimensional vector space with a canonical base {1, *i*, *j*, *k*} having the following Hamilton's multiplication rules:

$i^2 = -1 \qquad j^2 = -1 \qquad ij = k \qquad ji = -k$ (non commutativness)

Quaternion Multiplication table

x.y=	1	i	j	k
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

The base components *i*, *j*, *k* can be seen as the complex numbers; similarly a concept of conjugate can be defined:

Conjugate: with $Q = q_0 + q_1 i + q_2 j + q_3 k$ the conjugate is defined by $\overline{Q} = q_0 - q_1 i - q_2 j - q_3 k$.

Opposite: it is defined by $-Q = -q_0 - q_1 i - q_2 j - q_3 k$ Imaginary quaternion *:* when q_0=0.

Quaternion product: it is a distributive non-commutative product using the Hamilton's rules 1,*i*, *j*, *k* by the multiplication table.

For example $Q.\overline{Q} = (q_0 + q_1 i + q_2 j + q_3 k).(q_0 - q_1 i - q_2 j - q_3 k) = {q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2$

Other example: $\overline{Q_1.Q_2} = \overline{Q}_2.\overline{Q}_1$

There are no ambiguities with the definitions (or axioms) on a 4-dimensional canonical base, this is useful for fundamental algebra purpose, however this is quite heavy to use and there are no obvious meaning.

Second representation: $Q = \{q_0, \vec{v}\}$ where q_0 is the real part, and $\vec{v}$ is a vector having the components $\{q_1, q_2, q_3\}$ in the imaginary canonical base *i*, *j*, *k*. This base can however be regarded as equivalent to any geometrical base, for example the base of a 3-dimensional Cartesian frame. With this representation the link between quaternion and the real geometric world is more obvious.

Note*:* $\{0, \vec{v}\} = v = (Q - \overline{Q})/2$ *where here* v *is the imaginary quaternion corresponding to the vector* $\vec{v}$.

Quaternion product second form: $Q_1.Q_2 = \{q_0, \vec{v}\}.\{w_0, \vec{w}\} = \{q_0.w_0 - \vec{v}\cdot\vec{w}\ ,\ q_0\vec{w} + w_0\vec{v} + \vec{v}\wedge\vec{w}\}$ where "$\cdot$ "is the vector scalar dot product and "^" the vector cross product (right handed as usual) are used in the right hand side.

Note: *the product of two imaginary quaternion is simply :*$\{-\vec{v}\cdot\vec{w}\ ,\ \vec{v}\wedge\vec{w}\}$.

To get only the imaginary part of this quaternion (i.e. the cross product only), we shall consider the quaternion: $\{0, \vec{v}\wedge\vec{w}\} = (Q_1.Q_2 - \overline{Q_1.Q_2})/2$ *i.e.* $\{0, \vec{v}\wedge\vec{w}\} = (v.w - \overline{v.w})/2$.

Quaternion product matrix: *it can be done with standard matrix product "*"* $Q_1.Q_2 = [Q_1] * Q_2 =$ $\begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} * \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$

where $[Q_1]$ *is a skew symmetric matrix from* Q_1 *and with* Q_2 *written as column matrix.*

Also with writing $Q_1.Q_2 = \{w_0.q_0 - \vec{w}\cdot\vec{v}\ ,\ w_0\vec{v} + q_0\vec{w} - \vec{w}\wedge\vec{v}\} = [\hat{Q}_2] * Q_1 =$ $\begin{bmatrix} w_0 & -w_1 & -w_2 & -w_3 \\ w_1 & w_0 & w_3 & -w_2 \\ w_2 & -w_3 & w_0 & w_1 \\ w_3 & w_2 & -w_1 & w_0 \end{bmatrix} * \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$ *.The two matrixes are not identical because the product is not commutative* $[\hat{Q}_2] * Q_1 \neq [Q_2] * Q_1$.

Unit quaternion: Among all the quaternion, the ones used here are the unit quaternion: the norm being defined as $|Q| = Q.\overline{Q}$, a unit quaternion has a norm=1.

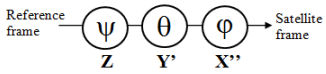
Quaternion inverse: $Q.Q^{-1} = Q^{-1}.Q = 1 \qquad Q^{-1} = \overline{Q}/|Q|^2$; for unit quaternion $Q^{-1} = \overline{Q}$.

Third representation for unit quaternion that are used now on, $Q = \{\cos\theta/2, \sin\theta/2\,\vec{u}\}$ where $\vec{u}$ is a unit vector. This can be written shortly with the following $Q(\theta, \vec{u})$. With this form, the quaternion represents an orientation change from a first frame (*f*) to a second frame (*s*): that is a rotation of angle

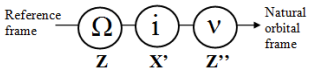
θ around $\vec{u}$ where $\vec{u}$ is considered as a vector having its coordinates written in the first frame (also as noted below $\vec{u}^{\,f}$).

Note: there is not a unique quaternion that represents an orientation change: other possibility is to consider the rotation of angle 2π- θ around the axis $-\vec{u}$: that is: a quaternion or its opposite produces the same orientation change. Thus when it is useful to keep the quaternion uniqueness for some orientation changes: a possible rule is to select the quaternion having a positive first component $\cos\theta/2 >= 0$ *and if not to use the opposite one.*

Orientation change: The meaning of a unit quaternion product $Q_1.Q_2$ is that an orientation change from a first frame to a third frame can be performed with $Q_1(\theta, \vec{u})$ and then with $Q_2(\phi, \vec{w})$ where $\vec{u}$ is a unit vector axis in the first (reference) frame and $\vec{w}$ is a unit vector axis in the second frame considered as its current reference frame. This makes obvious the correspondence between Euler angles or Cardan angles transformation and the corresponding quaternion. The advantage of the unit quaternion is that because they always have a norm of 1, the special cases of indetermination (gimbals lock) with the other transformations does not occur anymore. Thus for a whole quaternion $Q_{i,b} = Q_{i,o}.Q_{o,b}$ we can deduce $Q_{o,b} = Q_{i,o}^{-1}.Q_{i,b}$ where $Q_{x,y}$ is the quaternion from "x" to "y".

Example 1: 

the successive rotations from a first frame to a final one with the Cardan angles ψ,θ,φ around the axes Z (yaw), then Y' (pitch) then X'' (roll) in the Euler's sequence (3; 2; 1) *give the quaternion $Q_1(\psi,\vec{e}_Z).Q_2(\theta,\vec{e}_Y).Q_3(\varphi,\vec{e}_X)$.*

Example 2: 

the successive rotations with the Euler angles Ω,i,ν around the axes Z (precession), then X' (nutation) then Z'' (intrinsic rotation or spin) in the Euler's sequence (3; 1; 3) *give the quaternion $Q_1(\Omega,\vec{e}_Z).Q_2(i,\vec{e}_X).Q_3(\nu,\vec{e}_Z)$.*

Vectors: Considering a first inertial frame "*I*" and a second local frame for a body for example "*b*" having an instantaneous rotation $\vec{\Omega}$. One knows that the derivative of a vector $\vec{V}$ depends on the frame in which the derivation is performed. An abstract vector written with the letter $\vec{V}$ is a very concrete concept that does not depends on any frame. But operationally, the vector belongs to a vector space (here a 3-dimension), so that the frame in which one write its coordinates is of prime importance: every scalar product, cross product, matrix form product or local derivative shall be carefully performed within the same vector space i.e. within the same frame used to write the coordinates of the vectors. More explicitly let's use as superscript the frame in which the coordinates are written. $\vec{V}^{\,i}$ is the vector with coordinates written in the inertial frame; $\vec{V}^{\,b}$ is the same vector $\vec{V}$ but with coordinates written in the body frame and $\vec{\Omega}^{\,b}_{b/i}$ the instantaneous rotation of the body frame with respect to the inertial frame but with coordinates written in the body frame. It is obvious to say $\vec{V}^{\,i} \neq \vec{V}^{\,b}$, etc .

Vectors derivation: $\dot{\vec{V}}^{\,b}_{/i} = \left.\frac{d\vec{V}^{\,b}}{dt}\right|_{/i} = \left.\frac{d\vec{V}^{\,b}}{dt}\right|_{/b} + \vec{\Omega}^{\,b}_{b/i} \wedge \vec{V}^{\,b}$ where in $\left.\frac{d}{dt}\right|_{/x}$ indice $_{/x}$ stand for derivation reference frame.

Expression of vector after an orientation change with a quaternion: the **sandwich** product

The rule for transforming a vector $\vec{V}^{\,i}$ by an orientation change from "*I*" to "*b*" with a quaternion $Q = Q_{i,b}$ is given by the sandwiching product: $V'^{i} = Q.V^{i}.\overline{Q}$ where V^{i} and V'^{i} (without the vector arrow) are here the corresponding imaginary quaternion to the vectors $\vec{V}^{\,i}$ and $\vec{V}'^{\,i}$. A subtle but evident relation is to be mentioned: we consider the frame "b" that is the frame "i" after the rotation defined by the quaternion $Q_{i,b}$. Because the orientations change from "i" to "b" affect the whole frame, the coordinates of a vector before the rotation in the frame "i" and after rotation but in the frame "b" are

always the same, we have $V'^b = V^i$. Thus, we also have $V'^i = Q.V'^b.\overline{Q}$. Either this is true for any vector, thus $V^i = Q.V^b.\overline{Q}$. For example $\dot{V}^i{}_{/i} = Q\dot{V}^b{}_{/i}.\overline{Q}$.

Inversely, the other form of sandwiching product, $V^b = \overline{Q}.V^i.Q$ can be used to get the frame change coordinates matrix: $V^b = [R_{i,b}]*V^i$ with $[R_{i,b}] = [\overline{Q}]*[\hat{Q}]$

Derivation with respect to the time [R1], [R2] :

In the case of a quaternion $Q\,(=Q_{i,b})$ that represents the orientation change from the inertial frame to the body frame, Q depend on the time because the body frame is mobile. To represent the orientation change, the quaternion $Q(\theta, \vec{u})$ can be written with $\vec{u}$ a vector of the inertial frame base.

Using the 4-D canonical base, $\frac{dQ}{dt}_{/i} = \dot{Q} = \dot{q}_0 + \dot{q}_1 i + \dot{q}_2 j + \dot{q}_3 k$ *(here, quaternion base is inertial)*.

Derivation of quaternion, relation with instantaneous rotation:

From $V^i = Q.V^b.\overline{Q}$ we can say that $\dot{V}^i{}_{/i} = \dot{Q}.V^b.\overline{Q} + Q.V^b.\dot{\overline{Q}}$ because derivative $\dot{V}^b_{/b}$ is null, V^b being fixed in the rigid body "b". It follows $\dot{V}^i{}_{/i} = \dot{Q}.V^b.\overline{Q} - \overline{\dot{Q}.V^b.\overline{Q}}$. Also $\dot{\vec{V}}^b{}_{/i} = \vec{\Omega}^b_{b/i} \wedge \vec{V}^b$ from the vector derivation, when the vector $\vec{V}^b$ is as before constant (i.e. fixed in the rigid body frame "b").

This last cross product can be written in quaternion algebra, as seen before, $\dot{V}^b{}_{/i} = \left(\Omega^b_{b/i}.V^b - \overline{\Omega^b_{b/i}.V^b}\right)/2$ and also $\dot{V}^i{}_{/i} = \left(Q.\Omega^b_{b/i}.V^b.\overline{Q} - \overline{Q.\Omega^b_{b/i}.V^b.\overline{Q}}\right)/2$. Finally by identification, we set up a remarkable relation in the quaternion algebra: $\dot{Q} = \frac{1}{2} Q.\Omega^b_{b/i}$ where here $\Omega^b_{b/i}$ (without the vector arrow) represent an imaginary quaternion= $0 + pi + qj + rk$, that has the same coordinates of the vector $\vec{\Omega}^b_{b/i}$ written in the body frame. Further, $\frac{1}{2} Q.\Omega^b_{b/i} = \frac{1}{2}(q_0 + q_1 i + q_2 j + q_3 k).(0 + pi + qj + rk)$, can be performed, as seen before, with matrixes in the form "$Q_1.Q_2 = [\hat{Q}_2]*Q_1$", thus $[\hat{\Omega}^b_{b/i}] = \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}$ and $Q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$ give finally in matrixes: $\dot{Q} = \frac{1}{2}[\hat{\Omega}^b_{b/i}]*Q$.

Ref. [R1] Vernon Chi. "Quaternions and Rotations in 3-Space", 25 September 1998 ; Leandra Vicci, 27 April 2001

[R2] R. Guiziou., "DESS AIR & ESPACE, systèmes de contrôle d'attitude et d'orbite ", Uni. Aix-Marseille III, 2001.